(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 158 740 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.11.2001 Bulletin 2001/48

(51) Int Cl.[7]: **H04L 29/06**, H04L 12/56

(21) Application number: **00111191.3**

(22) Date of filing: **24.05.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

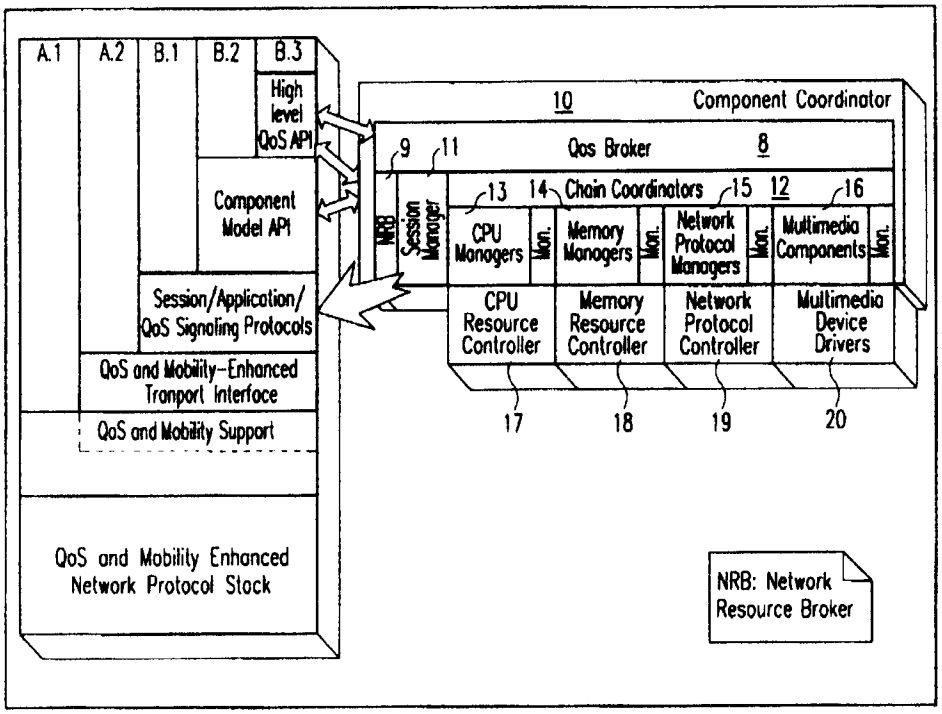
(71) Applicant: **Sony International (Europe) GmbH**
10785 Berlin (DE)

(72) Inventor: **Mandato, Davide**
c/o Sony Int.(Europe) GmbH
70327 Stuttgart (DE)

(74) Representative: **Körber, Martin, Dipl.-Phys. et al**
Mitscherlich & Partner
Patentanwälte
Sonnenstrasse 33
80331 München (DE)

(54) Quality of Service negotiation

(57) The present invention relates to a processing system and to pieces of software for one or more communication networks. Applications are provided with a platform- and network-independent framework for achieving cross-adaptability by providing components for QoS management in the communication network(s) by means of a component coordinator unit (10).



End System Reference Architecture

Fig. 3

EP 1 158 740 A1

Description

[0001] The invention hereby presented generally relates to the field of mobile multimedia middle-ware, computer networking, distributed processing systems, Quality of Service, handheld computers, web browsers, QoS brokers, QoS adaptation architectures, QoS negotiation protocols, and wireless communication.

[0002] Particularly, the present invention relates to a processing system for one or more communication networks and to pieces of software for one or more communication networks, the pieces of software being loadable in a memory means in one or more notes of the one or more communication networks.

[0003] The processing system for one or more communication networks according to the present invention is claimed in claim 1 and provides applications with a platform- and network independent framework for achieving cross-adaptability by providing components for QoS management in the communication network(s) by means of a component coordinator unit 10. The pieces of software for one or more communication networks according to the present invention are claimed in claim 21, are loadable in memory means of one or more notes of the one or more communication networks and provide applications with a platform- and network independent framework for achieving cross-adaptability by providing components for QoS management in the communication network(s) by means of a component coordinator unit 10.

[0004] Advantageous features are claimed in the respective subclaims.

[0005] Advantageously, the generic framework uses a platform- and network-neutral set of application adaptation mechanisms, including a QoS negotiation and re-negotiation protocol. In this case, the protocol may advantageously use piggyback mechanisms for negotiating and re-negotiating QoS parameters. Further advantageously, the generic framework bases on an application model in which each application is allocated to one of a set of application classes having different QoS levels. In this case, fallback mechanisms may be provided for a backward compatibility between the application classes. Advantageously, said generic framework bases on a modular progressive approach to address different types of applications which span from existing applications to envisioned more sophisticated applications that rely on middleware for achieving cross-adaptability. Further, the generic framework may base on a communication model with different functional communication levels for exploiting the various resources in a coordinated manner so as to achieve the desired overall QoS level. Hereby, the communication levels may include an application, a session, an association and a stream level.

[0006] Advantageously, a QoS broker unit is provided, which is managed by the component coordinator unit and coordinates local and remote resource management by using said negotiation and re-negotiation protocol. In this case, a network resource booker unit may be provided, which is directly coordinated by the QoS booker unit and manages network resource reservation mechanisms. Further, a session manager unit may provided, which is directly coordinated by the QoS booker unit for establishing and managing sessions in an implementation-independent way.

[0007] Advantageously, one or more chain coordinator units are provided, which are managed by the QoS broker unit and which manages one or more component chains, each chain being associated with a stream. Hereby, one or more CPU-manager units may be provided, which are coordinated by the chain coordinator units for managing CPU-resource usage. A CPU-resource controller unit may be provided, which provides the CPU-manager units with platform-independent resource status information retrieval and control.

[0008] Further advantageously, one or more memory manager units may be coordinated by the chain coordinator units for managing memory resource usage. A memory controller unit may provide the memory manager units with platform-independent resource status information retrieval and control. Further, one or more network protocol manager units may be coordinated by the chain coordinator units for managing network protocol resource usage. The network protocol controller unit may provide the network protocol manager units with resource status information retrieval and control.

[0009] One or more multimedia component units may be coordinated by the chain coordinator units for managing multimedia resources. Here, the multimedia controller may provide the multimedia component units with platform-independent resource status information retrieval and control.

[0010] This invention proposes a universal framework, upon which end-system architectures can be developed, providing QoS and mobility awareness to any kind of mobile multimedia and (within certain limits) legacy applications. More specifically, a technical solution (middleware) is hereby described, along with a meta-protocol for End-to-End QoS negotiation. These concepts have been designed so as to render applications elastic to QoS fluctuations and/ violations with respect to the given QoS contract.

[0011] The present invention is explained in more detail in the following description in relation to the enclosed drawings, in which

Fig. 1 shows a scheme of QoS adaptation framework according to the present invention,

Fig. 2 shows a scheme of coordinated adaptation versus control mechanisms according to the present invention,

Fig. 3 shows an example for an end system reference architecture,

Fig. 4 shows a scheme of the OSI application layer according to the present invention,

Fig. 5 shows an example of single and multiple association,

Fig. 6 shows a formal description of the proposed QoS adaptation framework according to the present invention,

Fig. 7 shows a logical structure of the negotiated information according to the present invention,

Fig. 8 shows a degradation path information exchanged during negotiation according to the present invention, and

Fig. 9 shows an example for association/stream set-up and hand-over scenarios according to the present invention.

[0012] The following table 1 gives the definitions of terms used in the present description.

Table 1:

Terminology	
Term	**Definition**
Adaptation	Mechanisms altering the application behavior so much as to compensate any fluctuation and/or even dramatic change in data transmission and/or processing quality, based on some user requirements specifying said quality. These mechanisms can be included in the application, or provided by some external entity (e.g. middleware).
Application	A computer program with a user interface. This invention addresses however with this term also computer programs acting as service providers, as e.g. a Video on Demand server. This kind of applications usually does not feature any interaction with users locally, except for a system administration interface.
Association	Relationship between two applications, providing the required reference structure for allowing said applications to cooperate.
Component	A prefabricated, customizable software unit featuring well-defined interfaces. Examples of components are Java-Beans, DCOM-objects, or CORBA-objects
Connection	Semi-permanent relationship between two nodes in a network; within the context of said relationship, data can be exchanged in an exclusive and controlled manner.
Deadline	A time limit, for the completion (or the fulfillment of a quota) of a task assignment.
Degradation Path	A list of alternative QoS profiles, each associated with a specific set of altered conditions with respect to the reference QoS requirements originally specified by the user. This list is used by QoS Brokers in order to degrade (or upgrade) the QoS in a controlled manner.
Device Driver	A piece of software that enables a computer to communicate with a peripheral device.
Differentiated Services, DiffServ, DS	Architecture providing different levels of service based on the differentiation of classes of Internet traffic. Rather than simply providing the same "best-effort" service to all network demands, Differentiated Services hopes to achieve a higher quality of service by differentiating between the needs of various predefined classes of service.

Table 1: (continued)

Terminology	
Term	**Definition**
Elastic Application	Application featuring adaptation, either through some built in capabilities, or thanks to some external means. An application capable of dealing with mutated networking and computing conditions in a controlled manner.
Framework	A set of correlated software units, embodying an abstract design for solutions to a number of related problems.
Hand Over / Hand Off	The mechanism of keeping a connection alive while one of the corresponding nodes is moving across interconnected network infrastructures.
Integrated Services, IntServ	The transport of audio, video, real-time, and classical data traffic within a single network infrastructure. Accomplished by reserving and allocating the proper amount of network resources to each given connection, based on the type of traffic that shall be carried over said connection.
Mobility, User	The ability of a user to access telecommunications services from different terminals and different networks, and the capability of the network to identify and authorize that user.
Mobility, Terminal	The ability of a terminal to access telecommunications services from different locations and while in motion, and the capability of the network to identify and locate that terminal.
Multimedia Service	A service in which the interchanged information consists of more than one type (e.g. video, voice, data, and graphics). Multimedia services have multi-valued attributes, which distinguish them from traditional telecommunication services such as voice or data. A multimedia service may involve multiple parties, multiple connections, the addition/deletion of resources and users within a single session.
Negotiation	Mutual discussions intended to produce an agreement with another party or parties. More specifically, QoS negotiation deals with the sorting out of a mutual set of QoS characteristics, as a tradeoff between each party's requirements.
Process	A part of a running software program or other computing operation that does a single task
Quality of Service, QoS	*The collective effect of service performance which determines the degree of satisfaction of a user of the service*
QoS Broker	Software unit orchestrating local, remote, and network resources, in order to provide and maintain a given level of QoS, as requested by all the cooperating parties. More specifically, the QoS Broker coordinates resource management across the various software and hardware units used within the system. To this extent, the QoS Broker manages QoS parameter translation, admission test, and negotiation. Therefore, the QoS Broker acts as an intermediary, dispensing cooperating entities from being aware of operational details concerning with other entities.
QoS Contract	Agreement between a user and a given service provider, specifying QoS requirements and constraints, as well as the policies required to keep track about QoS during all phases of said service.

Table 1: (continued)

Terminology	
Term	**Definition**
QoS Contract Type	Represents a QoS Category: it captures the structure of a class of QoS Contracts, by identifying how individual QoS Contracts specify the QoS properties over a given set of QoS parameter types (also known as *dimensions*). Each parameter type consists of a name and a domain of values. QoS specifications can be simply intended as a set of constraints over said domains, one per parameter type.
QoS Profile	Associates one or many QoS Contracts with a set of interface elements (methods) for a given interface.
QoS Violations	QoS Contract violations, which might occur even with service guarantees, because of resources shortage, e.g., network congestion.
Real-time OS	An Operating System featuring real-time capabilities. Real-time is a human rather than a machine sense of time, indicating the *level of computer responsiveness* that a user senses as sufficiently immediate or that enables the computer to keep up with some external vital process
Session	A multimedia session is a set of multimedia senders and receivers, and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session. Within the context of this invention, a session is an even more generic, not necessarily bound to multimedia concepts.
Stream/Flow	The sequence of information being transmitted from the source endpoint to the destination endpoint of a given connection
Task	The basic unit of programming that an operating system controls. Depending on how the operating system defines a task in its design, this unit of programming may be an entire program or each successive invocation of a program.

[0013] The following table 2 gives a list of abbreviations used in the present description.

Table 2:

Abbreviations	
AAA	Authentication, Authorization, Accounting
ACSE	Association Control Service Element
ACTS	Advanced Telecommunication Technologies & Services
AE	Application Entity
AEI	Application Entity Instance
AP	Application Process
ASE	Application Service Element
CASE	Common ASE
COTS	Commercial Off-The-Shelf
HW	Hardware
GUI	Graphical User Interface
IP	Internet Protocol
MACF	Multiple Association Control Function

Table 2: (continued)

Abbreviations	
MMU	Memory Management Unit
NRB	Network Resource Booker
PDA	Personal Data Assistant
QoS	Quality of Service
SACF	Single Association Control Function
SAO	Single Association Object
SASE	Specific ASE
SW	Software
UE	User Element
UML	Unified Modeling Language

[0014] As of this writing, following technologies are known, which are closely related to the topic hereby presented:

Sub-Topic	**Technologies**	**IP Owner Organizations and /or Companies**
QoS Broker *QoS Negotiation*	· *QoS Broker*	*Many academic-works have been published*
Context Negotiation	· *OSI Presentation Layer: Context Negotiation Protocol*	*ISO,* ITU-T (Rec. X.216 (07/94) INFORMATION TECHNOLOGY... OPEN SYSTEMS INTERCONNECTION... PRESENTATION SERVICE DEFINITION)

[0015] Table 3 lists the problems addressed by this invention. Each problem is identified by a *P_ID* identifier, later used for easy referencing the given problem item. Furthermore, a *Key Aspect* field captures the most meaningful aspect of the given problem item.

Table 3:

Lists of problems addressed by this invention		
P_ID	**Problem Description**	**Key Aspect**
1	Existing architectures do not take into account how *various* types of applications can manage end-to-end QoS in mobile scenarios.	Interoperability
2	Many of the existing architectures are *not modular*, rather full-fledged solutions	Modularity
3	Many of the existing architectures make some assumptions on the *network* resource reservation and allocation mechanisms	Configurability
4	The mechanisms dealing with application adaptation to mutated conditions of the network and of the computing environment (e.g. QoS violations, changes in resource utilization) have been addressed so far *without a clear distinction* of the various phases of applications and communications lifecycles.	Adaptability
5	Negotiation protocols have been so far defined as *proprietary* protocols.	Standardization

[0016] Different service interfaces can be identified above the OSI Transport Layer. Each interface serves a specific type of application. Table 4 describes each possible type of applications with respect to QoS and mobility aspects. The interface identifier indicated in the last column of said table is thereinafter used for referencing application types.

Table 4:

Application Taxonomy			
Legacy	Not QoS-aware	· Use sheer transport services. · QoS can be externally enforced - transparently to the applications - by manipulating packets (DiffServ marking) and/or explicitly reserving network resources (e.g. RSVP) through a configuration tool. Example: web browser	A.1
	QoS-aware	· Limited use of a QoS-Enabled Transport Service Interface, in conjunction with the aforementioned configuration tool.	A.2
QoS-aware	Self-managing	· Use existing (or soon-to-be-defined) OSI Session Layer protocols (e. g. SIP, H.323, RTP/RTCP) and QoS signaling protocols (e.g. RSVP) · Capable of performing DiffServ Marking · Feature adaptation logic for handling QoS violations · Directly handle multimedia resources (e.g. Frame Grabber, codecs). Example: Voice over IP clients	B.1
	Component -based	· Use abstract Session Layer Components, which can be mapped to any existing or future standard Session Layer protocols. · Can use prefabricated multimedia components. · Can use a framework (Component Coordinator-CC), which manages how components are pushed in / pulled out to/from a flow, and eventually reconfigured, in a dynamic manner. · Can include a proprietary full-fledged QoS Broker that (as one of its multiple functionality), can manipulate multimedia components through said CC. This functionality is however limited to the given application. Resource coordination with other applications can be made only indirectly, as long as proper low-level resource monitoring and control mechanisms are made available (e.g. through the OS). This is an envisioned kind of applications, not yet available on the market.	B.2
	Broker-assisted	· Use an external QoS Broker through a non-procedural language. · The QoS Broker can manage multiple applications, as opposed to the B.2 and (eventually) B.1 QoS Broker functionality, which is limited to one application only. · Users control the perceived QoS through a separate, application independent, QoS Broker GUI, either in an *entry level* or *expert* mode. · Template user profiles are made available to users, who can use them for creating their own profiles, either program-assisted (entry-level mode) or directly (expert mode). This is an envisioned kind of applications, not yet available on the market.	B.3

[0017] This invention focuses on architecture providing QoS and mobility awareness to all the aforementioned types of applications. More specifically, a technical solution (middleware) is hereby described, with respect to application types B.2 and B.3, and to the relationship of said application types with the other listed ones above. Table 5 summarizes the hereby-proposed solutions to the problems listed in Table 3.

Table 5:

Executive summary of claimed solutions		
P_ID	**Key Aspect**	***Solution Description***
1	Interoperability	Concerning the (re)negotiation process, the various types of applications described in Table 4 need to feature some form of backward compatibility, through fallback mechanisms. This invention describes such mechanisms in Table 6.

Table 5: (continued)

Executive summary of claimed solutions		
P_ID	**Key Aspect**	***Solution Description***
2	Modularity	Various types of applications can benefit from the proposed architecture, by using exactly the amount of support they need for accomplishing their tasks. Moreover, a component-based model allows applications to *reuse* commercially available multimedia components. One can also replace the latter with other components, based on mutated conditions (QoS violations) and/or cost factors (better quality/price ratio). See § Modularity.
3	Configurability	The proposed architecture is not tightly coupled to any particular network model. For a description purpose only, the given architecture is described with respect to an edge IP network scenario. Users' network packets enter/exit the network through an IntServ-based radio access network. Whereas the long haul traffic traverses as a DiffServ aggregate a core IP network (an Intranet or the Internet). The architecture can however be configured so as to use any available means for QoS signaling mechanism. To this extent, a Network Resource Booker (NRB) component is hereby defined. The configuration tool indicated in Table 4 with respect to application types A.1 and A.2 is basically a GUI for interactively managing said NRB component (which can be in any case used programmatically). See § Configurability.
4	Adaptability	This invention presents an application and communication models, which identify the major lifecycle phases of a multimedia application; each phase is associated with a specific adaptation mechanism. These mechanisms are organized in a framework. See § Adaptation mechanisms.
5	Standardization	This invention proposes the use of a platform- and network-neutral negotiation and re-negotiation protocol. Given that SIP features both these characteristics, the hereby proposed (re)negotiation protocol resembles said standard. See § Proposal of a Standard QoS Negotiation Protocol. A SIP binding is hereby described as well. See § Example.

[0018] As an option, each application class can be associated with a set of agents, for guaranteeing end to end coordination to the greatest possible extent. Each agent would be tailored to deal with each case of backward compatibility, as indicated in Table 6.

Table 6:

Backward Compatibility Mechanisms		
From	*To*	*Fallback Mechanism Description*
B.3	B.2	B.3 and B.2 QoS Brokers may be incompatible: in such a case, after a failed negotiation attempt, the peers will handle resources only locally. Otherwise the usual full-fledged negotiation process can take place. In the opposite direction, if B.3 applications cannot handle the negotiation process initiated by B.2 applications, resources are managed only locally.
	B.1	B.1 applications may feature no negotiation at all or a form of negotiation that is incompatible with the QoS Brokers assisting B.3 applications. In such a case, after a failed negotiation attempt from B.3, the peers will handle resources only locally. In the opposite direction, B.1 initiating applications may start some form of negotiation: if B.3 applications cannot handle this, resources are managed only locally.
	A.2	No negotiation takes place at all. In the case of a B.3 initiating applications, a negotiation attempt would fail. Resources are managed only locally.
	A.1	No negotiation takes place at all. In the case of a B.3 initiating applications, a negotiation attempt would fail. Resources are managed only locally.

Table 6: (continued)

Backward Compatibility Mechanisms		
From	*To*	*Fallback Mechanism Description*
B.2	B.1	B.1 applications may feature no negotiation at all or a form of negotiation that is incompatible with the QoS Brokers included into B.2 applications. In such a case, after a failed negotiation attempt from B.2, the peers will handle resources only locally. In the opposite direction, B.1 initiating applications may start some form of negotiation: if B.2 applications cannot handle this, resources are managed only locally.
	A.2	No negotiation takes place at all. In the case of a B.2 initiating applications, a negotiation attempt would fail. Resources are managed only locally.
	A.1	No negotiation takes place at all. In the case of a B.2 initiating applications, a negotiation attempt would fail. Resources are managed only locally.
B.1	A.2	No negotiation takes place at all. In the case of a B.1 initiating applications, a negotiation attempt (if any) would fail. Resources are managed only locally.
	A.1	No negotiation takes place at all. In the case of a B.1 initiating applications, a negotiation attempt (if any) would fail. Resources are managed only locally.
A.2	A.1	No negotiation takes place at all.

I. Modularity

[0019] The proposed architecture is modular with respect to two aspects:

- The architecture harmonizes various types of applications, which may or may not be compatible with each other. This modular *approach* allows developers to *progressively* deploy the given architecture.
- The middleware solution hereby proposed is centered on the concept of a *component model API*. The QoS Broker can be a component itself. Components can even be deployed at runtime (e.g. components developed as Java Beans can be downloaded from a remote repository). In order to deal with components, the hereby-proposed middleware is centered on the concept of a *Component Coordinator* (CC). The CC allows user applications to manipulate components and control their lifecycle (deployment, configuration, activation, deactivation, and disposal). More specifically, components can be "chained together", so as to combine their functionality in such a way information gets processed in a given, desired manner. Applications and/or the QoS Broker can modify chains even at runtime, so as to promptly respond to a given stimulus (e.g. change codec).

II. Configurability

[0020] The proposed architecture is configurable with respect to two aspects:

- The QoS-Enabled Transport Interface can be configured so as to map to specific QoS Signaling Protocols. E.g. a socket based QoS-Enabled Transport Interface can be configured so as to use RSVP and/or a DS Marker for resource reservation/allocation signaling. This configuration can transparently affect even type A.1 applications.

- Each component in the Component Model API can be individually configured. Applications type B.2 and the QoS Broker usually manage components configuration through the CC.

  - Concerning applications type B.3, the QoS Broker takes into account User Profiles information for configuring both component chains and lower-level entities (e.g. the QoS Signaling, network, and data link protocols).
  - The User's Profile information can be specified at high or low level of detail. In the former case, the QoS Broker shall perform *QoS parameter mapping* (i.e. translating user's high level requirements into a set of configuration parameters for each lower layer). In the latter case (expert mode), partial or even no QoS parameter mapping is required (depending on the expertise level of the user).
  - Users can also specify additional configuration information, also known as *degradation paths*. Each degradation path takes into account alternative QoS parameter configurations, to be used for addressing a given type of QoS violations in a *pre-planned* fashion.

- In order to facilitate users specifying their profiles, *templates* containing reference configurations can be made available. The QoS Broker therefore relies on a *User Profile Database*.
- A candidate implementation for *User Profile description language* could be QML Svend Frølund, Jari Koistinen, *QML: A Language for Quality of Service Specification*, Hewlett-Packard Laboratories Technical Report HPL-98-10 980210, Palo Alto, where QoS contracts are defined as *associated* to interfaces.

III. Adaptation Mechanisms

[0021] The following models are hereby proposed: Application Model and Communication Model. Individual adaptation mechanisms are described for each model. Aggregate adaptation mechanisms are finally presented at the end of this paragraph.

III.1 Application Model

[0022] This model focuses on the application lifecycle, with respect to local resources. Examples of resources are CPU, primary memory, network protocols, multimedia devices for each of the tasks that the application, as process, is composed of. Usage and QoS guarantees. As often pointed out in the literature, providing QoS in the network is not sufficient for guaranteeing that the users get the level of quality they ask (and pay) for.

[0023] Local resources are typically scarce in mobile computing devices like a PDA or a mobile phone. This scarcity can cause QoS degradation as soon as concurrent applications and/or simultaneous connections begin to compete for the limited amount of resources.

[0024] Basically one has to take into account that each new start of an application (thereinafter, an *entry*) may require the system to reshuffle resources so as to accommodate the new entry.

[0025] Each application run has a specific importance for the system and, ultimately, for the user. For example, this concept can be mapped to *priority* and *deadline* in a real-time OS for each task belonging to the given application. Therefore, the reshuffle process shall take into account the importance of the new entry with respect to the importance of the already allocated ones. This and other information can be associated with an application through an *application QoS contract*. This invention does not describe explicitly the QoS Contract content. The scope of this invention is in fact to provide a framework upon which specific solutions can be built.

[0026] This process can be based on a set of *aggregate importance classes*. These classes range from a (cheap) no-QoS-assurance class, up to an (expensive) premium-QoS-enforcement class.

[0027] Each application is assigned a given class, and will be tentatively assigned resources, according to the class-specified level of QoS.

[0028] Should resources not be available at admission test, applications belonging to lower classes would be assigned fewer resources until the new entry can be accommodated. This fallback (The term fallback applied to this process must not be confused with the similar term used for designating backward the compatibility mechanisms indicated in Table 6.) process begins from the class immediately below the one of the given application, and eventually continues (Other applications, with a higher or equal class compared to the one of the new entry, may have already been *degraded* to the given lower class. In such a case, the new entry is not entitled to seize any resource allocated to those applications. If no spare resources are available for the current class and all the allocated applications have a higher or equal class compared to the one of the new entry, the fallback process must be iterated further to the immediately below class.) to lower classes, until resources are finally made available to the new entry.

[0029] The same process applies to each application that has been degraded to a lower class. Applications belonging to the lowest class can be eventually preempted by applications with a higher class. Therefore this fallback process features a cascade mechanism.

[0030] The fallback mechanism can be triggered also by hand over events (for more details see following paragraphs).

[0031] The user has the possibility to control the aforementioned fallback mechanisms, by providing an application degradation path in addition to the application QoS contract. Such information would steer the fallback mechanism the way the user prefers.

[0032] For no reason, an application of given class can affect other applications of higher or equal class. Class promotions (and demotions) of applications, based on user preferences, are however possible. Proper AAA mechanisms shall regulate such user's requests.

III.2 Communication Model

[0033] Let us consider now a given application that wishes to establish a communication channel with one or more peers. The term peer is hereby used in a broad sense, so as to include even asymmetric client-server scenarios. Each

peer could be an application running either on the same, or on a remote-processing unit. Both cases can be addressed similarly, except that the latter deals with additional constraints concerning the usage of network infrastructure resources.

[0034] Generally speaking, applications can communicate with other peers within the high-level context of a session. As an example, a videoconference application would manage separately each conference in a session.

[0035] The session can then be broken into finer grained communication abstractions, the associations. Continuing the previous example, an association would represent a conference leg, i.e. the association between any two of the peers participating in said conference.

[0036] Finally, each association *can* be either mapped to a physical connection, or further decomposed in a set of information flows, or streams. This abstraction typically applies to multimedia applications.

[0037] Sessions, associations, and streams can be assigned each a QoS contract. The user can then pick up a specific configuration of the parameters indicated in template QoS contracts, and realize them into a corresponding set of QoS User Profiles (for more details, see § Adaptation Framework).

III.2.1 Session Establishment

[0038] The application can establish a session, provided that enough local and remote resources (Examples of session resources may be session priority, number of associated windows, estimated memory consumption for session control mechanisms.) are available. As an example, a small computing device may not support an unlimited number of sessions without running out of memory, or clogging up shared resources, like a display.

[0039] An adaptation mechanism similar to the one described in § Application Model can be applied to this phase as well.

[0040] Additionally, in order to check whether both peers are able to handle a given session, a negotiation process should take place. This invention proposes however a *lazy* approach: the session negotiation phase is deferred to the first association establishment (see § Association Establishment). This approach aims to reduce set-up time and network traffic. Please note that at session level, a negotiation process would be required *for each peer* participating in said session. Furthermore, the identity of said peers might not be known at session establishment phase.

[0041] In any case, the session has to deal with the coordination of session negotiation results originating from multiple association establishments: e.g. certain peers might not be able to manage a given session with the required QoS. The point is that each association might be either independent from- or tightly coupled to other associations. For instance: for the former case a videoconference, for the latter an online game or a distributed application. In the latter case, *application-specific policies* shall steer the overall negotiation process (e.g. suspend or even tear down the session, or simply degrade each association to a common QoS floor-level).

III.2.2 Association Establishment

[0042] This process translates into the establishment of one or more physical connections. Within a session, the application can establish an association, provided that enough local and remote resources, as well as network resources (This does not apply to the case of peers residing on the same computing unit.), are available.

1. *admission test* (CPU, multimedia devices, and primary memory): verifies whether enough resources are available locally. Similar

2. *fallback mechanism:* similar to the one described in § Adaptation Mechanisms, this mechanism is class-based.

3. *negotiation:* includes estimated protocol- (and, only for communication with remote peer, network-) resources usage, and a degradation path to be agreed upon. As indicated in § Session Establishment, *session negotiation* is included in this phase as well (during to the first physical connection establishment). This negotiation process takes also into account any eventual stream (Streams are assigned a QoS Contract as well as applications, sessions, and associations), which might be established within the given connection. If known a priori. For instance, a given association can be established with an audio, a video, and a data stream. This information is generally known at association establishment time.

4. *QoS Signaling* (only for communication with remote peer): once the two peers have come to a negotiation agreement, network resources can finally be requested, by using whatever QoS signaling mechanism is available (e.g. RSVP and/or DS marking).

[0043] The association establishment can be considered as subdivided into two phases: at first, best effort connec-

tivity is used for establishing the association and performing negotiation. Subsequently, as soon as the parties have agreed upon a common set of QoS, resources are allocated correspondingly to the desired QoS level. A similar mechanism applies in reaction to QoS violations.

III.2.3 Stream Establishment

[0044] Streams can be eventually created during the lifetime of an association, upon users' requests or as a consequence of a recovery from a temporary service disruption (see § Runtime and Hand-Over). In such a case, the admission test, negotiation, and QoS signaling mechanisms described in § Association Establishment shall apply. With the exception that no session negotiation would be required.

III.2.4 Runtime and Hand-Over Events

[0045] Following are the mechanisms that shall be used for managing QoS and mobility aspects within the context of a session.

1. QoS monitoring
2. QoS enforcement
    - 2.1. flow management (policing, shaping, shaping, synchronization, etc)
    - 2.2. resource control (e.g. acting on the network/cpu scheduler)
    - 2.3. application reconfiguration (e.g. modifying component chains)
    - 2.4. renegotiation (e.g. as a consequence of a Hand-Over event)

[0046] To this extent, this invention introduces the following distinction: *coordinated adaptation vs. control*.

III.2.4.1 Coordinated Adaptation

[0047] Coordinated Adaptation is any mechanism, which allows peers to recover from sensible mutated conditions (QoS violations and/or changes in resources usage) in a coordinated manner. More specifically, this implies the necessity of performing renegotiations. Coordinated Adaptation can thus be regarded to as a *horizontal* process.

[0048] This invention introduces the concept of a *renegotiation strategy (RS)*, based on degradation paths, that shall be agreed upon by peers at connection establishment time. The RS specifies a policy both peers shall follow when the conditions for renegotiations occur. Two policies can be identified:

- *Pre-planned sequence*: based on the information describing the mutated conditions (e.g. a QoS violation), both peers know how to degrade to a lower class of QoS in a similar fashion. The renegotiation process is thus implicit. Consequently, no network traffic needs to be originated for managing this process.

- *Index-based renegotiation*: the two peers can independently evaluate a lower QoS Class the given session should be degraded to. Renegotiation messages would be then exchanged, simply carrying the chosen set of references to the original degradation path items. This renegotiation process is thus explicit (i.e. requires some network traffic), but on the other hand the peers have more freedom in choosing the best degradation path for a given situation. For instance, fuzzy logic algorithms have been proven to be effective for reconfiguring QoS-related parameters in response to mutated conditions.

III.2.4.2 Control

[0049] Each peer shall take any possible action locally, in order to effectively and efficiently react to any mutated condition in the local computing unit and/or in the network, before ever relying on any specific adaptation mechanism. Control can thus be regarded to as a *vertical* process.

[0050] Given that the set of QoS parameters can be globally described as an element in a given n-dimensional space, the peer can use any control theory based mechanism, forcing said element to stay within a given region centered on the element.

[0051] This can translate for instance in the fallback mechanism described in § Application Model, and/or reconfiguring component chains (see § Runtime and Hand-Over Events).

III.3 Adaptation Framework

[0052] The overall adaptation framework is depicted in Figure 1, by following Profile 5a-5d, Contract 6a-6d, and Contract Type 7a-7d concepts, as well as the extended UML graphical notation.

[0053] QoS Profiles 5a-5d specify the binding of a given QoS contract 6a-6d (which is problem domain-specific, but implementation-independent) with a given interface. In this case, the interfaces are QoS Management interfaces as supported by the application itself and/or by the middleware.

[0054] As partially indicated in the previous paragraphs, applications 1 may be required to perform session 2 coordination, sessions 2 may be required to coordinate associations 3, and associations 3 may need to synchronize streams 4.

[0055] This recursive dependency scheme reflects onto the various QoS Profiles 5a-5d. More specifically, metrics and policies will be needed: the former for specifying synchronization constraints (e.g. maximum tolerable lip-sync delay), the latter for indicating which type of actions should be taken when the synchronization is lost (e.g. automatically recover or ask user intervention).

[0056] Figure 2 summarizes in a more organic manner the various mechanisms so far described, putting in evidence the distinction between coordinated adaptation and control mechanisms.

[0057] More specifically, the figure indicates that the fallback mechanism described in § Application Model applies not only to the Application 1, but also to the Session 2, Association 3, and Stream 4 abstractions. The reason is that each of the aforementioned abstractions uses local resources in various (implementation dependent) manners: e.g. session 2 may be associated to a certain number of tasks, each demanding some CPU and memory resources.

[0058] In order to be effective, the admission test and fallback mechanisms can thus be performed in a *hierarchical* manner, where at step *n* an estimated (or default) number of instances of the *n+1* abstraction are taken into account. This *distributed* approach guarantees that resources are effectively committed *only* if a minimum, meaningful functionality can be offered to the user.

[0059] For instance, it would not make any sense to start a videoconference in a resource-constrained mobile device, if no resources are available for at least one connection.

[0060] This form of *booking* mechanism is also meant to speed up the establishment of connections.

IV. End System Reference Architecture

[0061] Figure 3 provides a graphical representation of the application taxonomy presented in Table 4, with respect to existing protocols and the hereby-proposed middleware (see Table 7).

Table 7:

Description of the Software Units and Components	
Unit Name	**Unit Description**
QoS Broker 8	Component providing applications with the highest level of QoS awareness. The QoS Broker 8 basically manages any local and remote resource as a sort of OS extension. To this extent, all the applications running on a given end system will then be using and/or influenced (either indirectly - as in the case of applications type A. 1 up to B.1 - or directly) by such component. The QoS Broker 8 maps QoS parameters throughout the middleware, the protocol layers, the OS, and any multimedia component that might been used. The mapping is bi-directional, i.e. from User QoS Profiles down to low-level tunable firmware, and vice versa. Furthermore, the QoS Broker 8 implements the aforementioned adaptation mechanisms, delegating however low-level tasks to other specific components. The QoS Broker 8 offers applications with a high level QoS interface, thus enabling them to request services from lower layers, along with given high-level QoS User Profiles. In particular, the QoS Broker 8 manages the negotiation protocol described in § Proposal of a Standard QoS Negotiation Protocol. In order to be implementation independent, the QoS Broker 8 uses Session Manager component 11, which abstracts session level details (e.g. the use of H.323 protocols or SIP).

Table 7: (continued)

Description of the Software Units and Components	
Unit Name	**Unit Description**
	In order to deal with QoS signaling, the QoS Broker 8 uses the Network Resource Booker component 9, which abstracts network resource reservation and allocation mechanisms (e.g. the use RSVP and/or DS in IP based solution). The QoS Broker 8 manages multimedia components (see below), by configuring component chains. Should adaptation been required, the QoS Broker shall reconfigure said chains and/or even substitute some components within the given chains.
Component Coordinator 10	This software unit provides applications with a generic framework for managing components. More specifically, this unit 10 deals with component and component-chain lifecycle (deployment, activation, management, deactivation, and disposal). A component-chain can be managed as a whole by the Chain Coordinator component. During the deployment phase, this unit 10 can even download components from remote repositories (e.g. CORBA or Java Beans components). To this extent, this unit abstracts all the platform specific mechanism for locating and downloading said components.
Chain Coordinator 12	This component 12 manages one or many component chains in order to guarantee flow synchronization within the tolerances requested by the user. Furthermore, this entity concentrates QoS events being generated by the monitors associated with each component, in order to provide the QoS Broker 8 with refined and concise information.
Session Manager 11	This component 11 coordinates sessions and abstracts any session level detail (e.g. the use of H.323 protocols or SIP). The session manager 11 is responsible for the Session (see § Communication Model) establishment and management. In particular this component 11 shall report to the QoS Broker 8 any QoS violation signal, generated by the lower protocol layers, which might occur during the lifetime of a session (e.g. a Hand-Over event)..
Network Resource Booker 9	This component 9 abstracts network resource reservation and allocation mechanisms (e.g. in IP based solution, a RSVP daemon and/or a DS marker). This component 9 can be used standalone for application types A.1, A.2 (by using QoS Signaling transparently to the application), and B.1.
CPU Manager 13	Each instance of this component 13 manages CPU (local) resource usage for a given flow. More specifically, this component enforces flow-based QoS control mechanisms like flow policy, shaping, monitoring, etc. Monitors are put in evidence in Figure 3.
Memory Manager 14	Each instance of this component 14 manages primary memory (local) resource usage for a given flow. More specifically, this component enforces flow-based QoS control mechanisms like flow policy, shaping, monitoring, etc. Monitors are put in evidence in Figure 3.
Multimedia Component 16	Each component 16 consists of either self-contained software entities (like a QoS filter) or software drivers for a specific device (e.g. a frame grabber). Monitors are put in evidence in Figure 3.
Network Protocol Manager 15	Each instance of this component 15 manages network protocol (local) resource usage for a given flow. More specifically, this component 15 enforces flow-based QoS control mechanisms like flow policy, shaping, monitoring, etc. Monitors are put in evidence in Figure 3.

Table 7: (continued)

Description of the Software Units and Components	
Unit Name	**Unit Description**
CPU Resource Controller 17	A per-computing-unit centralized software unit 17, offering the aforementioned CPU Managers with fine CPU control and monitor. This unit 17 is platform dependent, but communicates with said managers through a platform independent interface. More specifically, this unit allows CPU scheduler management (e.g. changing quanta and/or deadlines assigned to each task).
Memory Controller 18	A per-computing-unit centralized software unit 18, offering the aforementioned Memory Managers 14 with fine CPU control and monitor. This unit 18 is platform dependent, but communicates with said managers through a platform independent interface. More specifically, this unit allows fine primary memory management (e.g. MMU management).
Multimedia Controller 20	A per-computing-unit centralized software unit, offering those Multimedia Components 16 that deal with a specific hardware multimedia device, with fine device control and monitor. This unit is platform dependent, but communicates with said managers through a platform independent interface.
Network Protocol Controller 19	A per-computing-unit centralized software unit 19, offering the aforementioned Network Protocol Managers 15 with fine control and monitor mechanisms over the protocol stack. This unit 19 is platform dependent, but communicates with said managers through a platform independent interface. More specifically, this unit allows fine lacal network resources management (e.g. packet buffer size or packet scheduling management). This unit can be interfaced to a data link / physical protocol layers manager unit, in order to allow fine parameter tuning and control over said protocol layers from within the QoS management plane. To this extent, the QoS and Mobility Enabled Transport interface indicated in Figure 3 would then redirect QoS-related information/operation to the QoS Management plane and, through the Network Protocol Controller, to the data link / physical protocol layers manager unit. This unit is not explicitly indicated in Figure 3. It is envisioned that the interface between the Network Protocol Controller and the data link / physical protocol layers manager unit will be standardized in the next future.

V. The proposed QoS Adaptation Framework from an OSI perspective

[0062] The purpose of this paragraph is not to rigorously map the hereby-presented framework to the full-blown OSI architecture, rather to indicate the *analogies* between the concepts presented so far, and some key aspects of the OSI model.

[0063] Purpose of the OSI standard is to define the *information exchange* among *open systems*. Systems that can freely cooperate in order to achieve a common objective. Key aspects of open systems are interoperability and application portability. To this extent, open systems negotiate a set of common functionality so that they can share a common view of each other, thus hiding any internal characteristic peculiar of each system.

[0064] As background information, following please find briefly summarized some OSI concepts concerning with the Application, Presentation, and Session Layers, with the focus being only on those aspects, which the author considers as essential for formalizing this invention.

V.1 The Application Layer in the OSI Model - Background Information

[0065] Figure 4 presents a high level view of the OSI Application Layer. This layer is the topmost one in the 7-layer OSI architecture. As such, this layer offers a service directly to the final user, i.e. the *Application Process* (AP). The Application Process 20 is actually outside the scope of the OSI: within the OSI model, an Application Process is represented by an *Application Entity* 21 (AE). An AE 21 is defined as a set of:

- A *User Element* 22 (UE), which represents the AP 20 to the AE 21, therefore being the joining element between

the real system and its representation as open system.

- A collection of logically correlated *Application Service Elements* 23, 24 (ASE) each defined as a functional module.

[0066] The OSI Application Layer thus provides said AEs with functionality for supporting communication at Application Process level. This functionality is carried out through protocols designed for specific applications and/or general-purpose routines.

[0067] The AE nature is determined by the choice of the ASEs being used. These elements can not exist by themselves. Therefore, an application entity is composed of one or more ASEs, each accomplishing a specific task on behalf of the user application. As a consequence, the architecture of the application service differs considerably from the architecture of the other OSI layers; in fact, the latter can be represented by an entity that offers a particular service, while the application layer can be represented through the set of ASEs and *negotiable* options.

[0068] More specifically, each Application is characterized by a set of *Specific ASEs* (SASE), ad hoc defined and offering functionality that are typical of the application they have been designed for (e.g. FTAM manages file transfer functionality).

[0069] Furthermore, any functionality that is common to multiple types of applications can been abstracted out. The corresponding ASEs are called *Common ASE* (CASE). In particular, any ASE offering services to another ASE can be considered as a CASE.

[0070] From a procedural perspective the cooperation among application processes takes place thorough logical bounds, called *Application Association*, established between each couple of AE residing in different processing units. The AE originating an association is called *Association Initiator*, whereas the other peer is called *Association Responder*. Therefore, an Application Association is the relationship between two AEs, and offers the required reference structure for allowing them to cooperate.

[0071] The procedures for establishing associations include the exchange of proper *Application Protocol Data Units* (APDU), by using the services offered by the OSI Presentation Layer. Said APDUs also conveys control information, which the AEs can use for managing the Application Association and coordinating their functionality.

[0072] For each Application Association, the way information is exchanged needs to be defined: to this extent, the set of procedures used between the AEs is called the *Application Context* of the given Application Association. An Application Context is whatever type of information AEs require for cooperating over a given Application Association. In particular, said context would always include the description of the set of ASEs used by each AE involved in said association.

[0073] To this extent, in order to ease the definition of processes that concur in setting up the communication between AEs, some *abstract models* have been identified, each logically grouping functionality that have already been defined for a given Application Context. The *Single Association Object* (SAO) is the abstract model of the functions corresponding to a specific application association. It is composed of one or more ASEs, among those belonging to the given AE.

[0074] Since the ASEs belonging to a SAO are those that are part of the Application Association represented by the SAO, one of them must be the *Application Control Service Element* (ACSE). This ASE deals with the establishment, release, and abort of Applications Associations.

[0075] Within each SAO one can identify the procedures that regulate the interactions among ASEs, and those which regulate the interactions among said ASEs and the OSI Presentation Layer. The *Single Association Control Function* (SACF) is the abstract model that identifies the set of said procedures; it represents the execution of rules defined within the application context, and therefore its description is implicitly contained within the Application Context and, in particular, in the ASEs descriptions.

[0076] If the Application Entity opens more associations, multiple SAOs can be correspondingly identified. These SAOs can feature different combinations of ASEs. A *Multiple Association Control Function* (MACF) can be used if some form of coordination is required among some or all the Application Association of a given AE. Substantially, the MACF governs le interactions among a group of SAOs within a given AE, over some associations. That is, the temporal sequence of events within the various associations, the relationships among them, and everything that could be useful for the management of a multiple association.

[0077] Figure 5 depicts the *use* of AEs (i.e. Application Entity Invocations); it shows two Application Associations, one between two instances of SAO 1 and the other between two instances of SAO 2. In both cases, the MACF controls the use of services offered by SAO 1 and SAO 2 in AEI 2. Even though not indicated in the figure, direct associations without the mediation of any MACF are of course also possible.

[0078] Examples of ASEs are:

- ACSE (Association Control Service Element): deals with the establishment, release, and abort of Applications Associations.

- FTAM (File Transfer Access and Management): deals with file transfer functionality in the OSI context. Historically, this has been one of the first ASEs ever developed, and it is one of the most interesting applications.

- RTSE (Reliable Transfer Service Element): offers the procedural elements for the reliable transfer of information. This functionality mirrors the OSI Session Layer one, but offers the possibility of using ASN.1 rather than byte formats thus simplifying the complexity of the Session Layer.

- ROSE (Remote Operation Service Element): allows the support of event those distributed applications, which are not strictly based on the client-server model.

- CCR (Commitment Concurrency Recovery control): designed for managing distributed applications, by solving consistency and concurrency problems that might occur in such cases.

- CMISE (Common Management Information Service Element) and SMASE (System Management Application Service Element): they both deal with network management in the OSI context. Both rely on ROSE.

V.2 The Presentation Layer in the OSI Model - Background Information

[0079] The OSI Presentation Layer establishes, manages, and terminates communication sessions between OSI Application Layer entities. The OSI Presentation Layer main purpose is to render virtual any syntactical difference (Whereas the OSI Application Layer main purpose is to render virtual any semantic difference among applications in an open system. This means to ensure that said applications share a common dialogue context, that is *what* to discuss about and *how* to do that. A common semantic implies the use of a common syntax.), which might exist among the various applications cooperating in an open system context. Another key feature of this Layer is *data encryption*, which is, however, outside the scope of this writing. Encryption can be accomplished theoretically in any of the seven OSI layers, but actually only the Physical, the Session and the Presentation Layers are considered by some authors as the most suitable ones. This goal can be achieved by either using common transfer syntax, or a common external representation of data.

[0080] The concrete data structures in the local syntax are mapped to data types, which describe them in terms of an *Abstract Syntax*. An example of Abstract Syntax is the ASN. 1 standard. A *Transfer Syntax* is then applied to the Abstract Syntax, in order to obtain the *value* corresponding to a given abstract data type (serialization).

[0081] The couple (*Abstract Syntax, Transfer Syntax*) is called *Presentation Context*, which is defined by the Presentation Layer for implementing its functionality. Actually, this Layer supports two types of contexts:

- A *Defined Context,* which is the one agreed upon among the two users and the Presentation Layers, through a negotiation process.

- A *Default Context*, which is always available at the Presentation Layer. This context is used if the Defined Context has not been agreed upon.

[0082] In order to expedite the negotiation process, each Presentation Context is assigned *a Presentation Context Identifier* (PCI), an integer value that can be used for later referencing a specific context.

[0083] For instance, an AE can be composed of two ASEs. Each ASE may define its own abstract syntax for communicating with the peer ASE. Generally, these abstract syntaxes can differ. In order to avoid confusion, two distinct Presentation Contexts, one per ASE pair, are set up during the presentation connection establishment. This distinction is achieved by marking each context with a specific PCI.

[0084] The Presentation Layer mandates a set of services and protocols dealing, among other functionality, with the Presentation Context *negotiation* (at presentation connection establishment time) and *renegotiation* (during the connection lifetime) process.

[0085] At connection establishment time, a Defined Context Set (DCS) is negotiated. The Initiator first specifies in said DCS a list of all the Presentation Contexts that it can handle. For each context, the PCI and the Abstract Syntax are specified. The Initiator does not specify anything concerning with the Transfer Syntax. The OSI Presentation Layer, used by the Initiator, in fact provides this information. The presentation entity adds *all* the possible Transfer Syntaxes that are compatible with a given Abstract Syntax listed in the DCS. If no Transfer Syntaxes are available for a given Abstract Syntax, an exception is raised to the application process, and the corresponding item is removed from the DCS.

[0086] The complete DCS information is then sent to the peer OSI Presentation Layer, which checks whether the specified Transfer Syntaxes can be supported, and then forwards to the local user (the *Responder*) the following information for each Presentation Context listed in the DCS:

- PCI
- Abstract Syntax
- A flag indicating whether a particular Presentation Context has been accepted or not (*acceptance* or *provider rejection*).

[0087] Should a given context be rejected, one of the following reasons are specified:

- Unspecified Reason
- Unsupported Abstract Syntax
- None of the proposed Transfer Syntax has been accepted
- Reached DCS Local Limit

[0088] The Responder then replies to its presentation service provider, which Presentation Context the Responder is able to support. The Presentation Layer communicated this information back to the originator peer, in terms of:

- PCI
- Abstract Syntax
- Flag: acceptance/provider rejection indicator
- The chosen (unique) Transfer Syntax

[0089] This information is forwarded to the Initiator and the negotiation is over. The DCS contains now only those Presentation Contexts (each with its own unique Transfer Syntax) that both peers can support.

[0090] This process can be repeated at any time, during the lifetime of a presentation connection.

V.3 The Session Layer in the OSI Model - Background Information

[0091] The OSI Session Layer establishes, manages, and terminates communication sessions between OSI Presentation Layer entities. More specifically, the Session Layer main purpose is to structure and regulate the data flow among users, managing any failure and/or exceptional event that might occur. In order to deal with the latter aspects, the session service relies on the *checkpoint* concept. A checkpoint is a logical mark inserted during the lifetime of a connection between two users. When a failure/exception occurs, provisions are made available for recovering from the given event by rolling back to one of the connection states marked with a checkpoint.

[0092] The OSI Session Layer can map *session connections* to *transport connections* (offered by the underlying OSI Transport Layer) in different ways: one-to-one correspondence, one session connection over multiple transport connections, multiple session connections over one transport connection. Transport connections can be *reused* for multiple session connections, or *new* ones can be established for given session connections. *This transport connection assignment is based on the users' QoS requirements.*

V.4 A formal description of the proposed OoS Adaptation Framework

[0093] Figure 6 presents the QoS Adaptation Framework described in § Adaptation Framework, conforming to the OSI concepts indicated in § The *Application Layer* in the OSI Model - Background Information, The *Presentation Layer* in the OSI Model - Background Information, and The *Session Layer* in the OSI Model - Background Information. The following formal description refers to the full-fledged middleware supporting application type B.3. The same concepts can be applied, *mutanda mutandis*, to the application types B.1 and B.2.

[0094] The QoS Broker 30 (i) implements the main QoS adaptation logic and (ii) coordinates the various entities (e. g. the management of applications; the coordination of associations, sessions, and streams). The QoS Broker is a component itself and as such, a central Component Coordinator SW Unit manages it. This Unit manages also all the other components that fit into the architecture. More specifically, the QoS Broker 30 relies on an external component, the NRB 31, which however users can otherwise *directly* access through a specific GUI (this applies to any type of applications - with only some limitations for applications type A.1 and A.2, where no QoS *signaling* is available, but QoS configuration is allowed).

[0095] The Application Process 32 (AP) represents the user application. The AP 32 can use the middleware, but the middleware only indirectly affects it. The middleware in fact performs QoS adaptation by tuning the AP resource usage, as described in § Application Model. To this extent, the AP 32 is associated with an AE instance 33 (AEI). The association is actually realized through an UE 34, which represents the AP 32 within the middleware. The UE 34 contains information concerning the AP 32, like priority, class (see § Application Model), and (if any) configuration information (for instance some memory requirements) that might be useful for performing admission test and the fallback mecha-

nisms. The UE 34 also coordinates a list of open sessions (for the sake of simplicity, the figure represents only one session), under QoS Broker 30 direct control.

[0096] Each session, as described in § Communication Model, can be split in several associations. The set of said associations is coordinated by the Session Manager 35, which therefore acts as a MACF, under QoS Broker direct control.

[0097] Each association is represented by a SAO 36, 37: the Chain Coordinator 38 (see § Adaptation Framework) acts as a SACF, by coordinating ASEs (like multimedia components and Resource Managers - see § End System Reference Architecture), under QoS Broker direct control. More specifically, the Chain Coordinator 38 handles multiple chains each eventually associated with a given stream. To this extent, ASEs are instantiated per stream. Furthermore, certain components can be implemented as stubs, which represent a high level interface of the given components. This level of indirection introduces polymorphism property in the architecture: the actual component implementations (skeletons) are typically handled within the OSI Presentation Layer as a sort of component skeleton library. In most of the cases, in fact, multimedia components deal with typical OSI Presentation issues, like e.g. encoding, compression, and encryption functionality. On the other hand, multimedia components that cannot be modeled in this way are e.g. those which deal with multimedia device drivers. Which skeleton will be actually used, depends on the result of an *extended* Presentation Context *negotiation* process, as described below.

[0098] To this extent, the present invention slightly differs from the OSI model: the QoS information is *negotiated* beforehand at the highest layers, and only later *communicated* to the OSI Transport and lower layers, in terms of negotiated QoS requirements specific to the physical connections that have to established.

[0099] Being implicitly included in the OS context, the Resource Controllers are therefore not indicated in Figure 6 for simplicity reasons.

V.4.1 Extended Presentation Context Negotiation Process

[0100] The idea is to leverage the existing Presentation Context Negotiation Process for exchanging additional context information among peers. More specifically, in addition to the Abstract Syntax (which might in certain cases do not even apply - the OSI model is hereby used only as a reference model, and not all of its concepts are hereby enforced), the peers now will exchange QoS information, in terms of:

- List of component stubs to be used
- Application QoS Profile
- Session QoS Profile (if any session is to be used, assumes one session will be used by default)
- Association QoS Profile (assumes one association will be used by default)
- Stream QoS Profile (if any stream is to be used, assumes one stream will be used by default)

[0101] The Association and Stream QoS Profiles contain estimated information concerning protocol stack resource usage. These estimates are computed through the Resource Managers dedicated to each flow and resource type, in conjunction with the centralized services offered by the resource controllers. QoS Brokers shall then tune these estimates during the runtime, using the adaptation mechanisms described in § Adaptation Mechanisms.

[0102] The ASE information is organized in a DCS: a PCI, an Abstract Syntax (if any required), and a Component Stub (if any needed) are assigned to each ASE that needs to establish a communication with a peer one. For instance, the initiator may choose to use a data compressor with a given compression ratio: given that the two peers may have various brands/flavors of compressors. These compressors might even be not fully compatible with respect to each other. Therefore, the peers need to agree (i.e. negotiate) upon which actual compressors (i.e. Skeletons) they shall use. Being simply piggybacked by QoS Presentation Layer protocol data units, the QoS Profiles are placed in front of the DCS. In order to correlate session, associations, and streams, each Profile is associated with an identifier, as indicated in Figure 7.

[0103] For the sake of simplicity, this figure represents the specific case of one session, one association, one stream, and one ASE description. More generally, each of these descriptions can be replicated according to the actual entities been active, so as to group negotiation information in one data structure. This typically applies to ASEs, but can be extended as well to the other abstractions.

[0104] The Presentation Layer serving the Initiator shall complete the negotiation information to be sent to the peer, by selecting from the libraries the Transfer Syntaxes (see § The *Presentation Layer* in the OSI Model - Background Information) and the Component Skeletons that comply respectively to the specified Abstract Syntax and Component Stubs.

[0105] At the peer side, the OSI Presentation Layer shall validate the Transfer Syntaxes (if any) and Component Skeletons (if any), and relay the resulting information to the Responder. The Responder is shepherded by the peer QoS Broker: admission test and fallback mechanisms (see § Application Model) are performed at the various layers

(including the lower layers), and a response is handed back over the Initiator accordingly. This scheme is based on the same mechanisms described in § The *Presentation Layer* in the OSI Model - Background Information.

[0106] The negotiation process can be repeated in case the Initiator is not satisfied with the Responder feedback. In every step, the resources are allocated along the path, as indicated in Figure 9 (see § Proposal of a Standard QoS Negotiation Protocol for a detailed description of this figure).

[0107] The same mechanisms can be applied to the renegotiation process, the main difference being that the communication links among peers are already established.

[0108] What presented so far reflects an ideal world case, where a specified set of QoS requirements can be specified once forever, and the QoS Brokers are able to satisfy said requirements by simply tuning the overall systems in a coordinated manner.

[0109] In the real world, however, sudden QoS violations can be so dramatically big to drive the QoS Brokers offset. In such a case, proper degradation path information (see § Adaptation Mechanisms) shall be included in the negotiation information. This additional information can be structured as additional sets of information, in the same way as indicated in Figure 7 (which represents the reference context). In order to distinguish items belonging to the degradation path from the ones belonging to the reference context, each item (the *description* in Figure 7) shall assigned an ID, *mathematically correlated with the corresponding item in the reference context*. The mathematical relationship could be a proper hierarchical *bit mapping* scheme, or any other solution. This invention does not mandate any specific relationship, as this is seen to be an implementation dependent issue. Figure 8 presents a sample case of degradation information applied to the negotiation information.

[0110] As indicated in § Adaptation Mechanisms, the renegotiations can thus be limited to a simple item ID exchange, and only when the two QoS Brokers do need to synchronize. By default, in fact, the QoS Brokers are assumed to be able to use degradation path information at the unison. Only if, after a configurable timer has expired, the monitored QoS is still violating the (degraded) QoS requirements, the QoS Brokers initiate a renegotiation process, where only the sets of chosen items IDs are negotiated (given that the corresponding information has already been negotiated at connection establishment time).

[0111] Only in the case one of the peers has got corrupted the previously negotiated information, or whenever the user has modified her/his configuration information, an explicit renegotiation process (similar to the one described for the reference negotiation case) shall apply. A QoS Broker receiving such an explicit request shall automatically be forced to discard previous information and complete the renegotiation process, as if it was a new negotiation process.

V.4.2 A final note about the OSI Presentation Layer

[0112] So far the proposed QoS Adaptation Framework has been described with respect to an OSI Presentation Layer, with augmented negotiation capabilities. This concept has been introduced for formalizing said framework, according (to a certain extent) to the OSI model.

[0113] From an implementation point of view, however, the OSI Presentation Layer functionality is partly included in COTS Session protocol implementations (like SIP) and partially in the QoS Broker Component. More specifically, the QoS Broker Component can be thought of as composed of a set of four major components:

- QoS Profiles Database Management System
- QoS Parameters Mapper
- QoS Control/Coordinated Adaptation Finite State Machines
- QoS (re)Negotiation Engine

[0114] The latter component is dealing with all the logic behind the negotiation *process*, whereas session protocols like SIP can be used for implementing the negotiation *protocol*.

V.4.3 User interaction

[0115] The negotiation process has been described so far as a completely automatic mechanism. Indeed, the QoS Broker offers a high level of automation. Nevertheless, the Responder can agree with the Initiator upon establishing any given session, association, and stream, provided that human control is exerted at some level.

[0116] The Responder could in fact be an instance of configured automata (i.e. a Video on Demand server), or just a human picking up a call (e.g. a videoconference application). Even in the latter case, however, the user can program the QoS Broker to take into account details (such as QoS related issues), thus being able to take high level decision (accept / reject / modify the call request).

VI. Proposal of a Standard QoS Negotiation Protocol

[0117] This invention proposes the standardization of a *generic* QoS Negotiation Protocol: this protocol is not tight to any particular network technology. Actually, rather than proposing yet a new full blown protocol, this invention describes a meta-protocol, which can be implemented by mapping the procedures and the information elements described below, to existing protocols like, e.g., SIP.

[0118] To this extent, *generic logical* messages (thereinafter indicated as *methods*), like *association establishment, stream establishment, acknowledgements*, and *renegotiation*, are widely used in order to represent the negotiation signaling between the peers, which is therefore performed *in-band*.

[0119] These generic messages shall be mapped to the protocol specific analogous messages. Provided that the selected protocol specification allows extensibility features. Otherwise, proper actions shall be taken within the proper standardization bodies in order to explicitly support the information elements hereby-described. The important aspect of this invention is that real-protocol messages *piggyback* the negotiation information (including degradation paths), along with the usual information. This *lazy* approach implies that no explicit negotiation messages are required whatsoever, thus reducing the complexity of the protocols and the setup (or recovery, as in the case of renegotiations) delay.

[0120] Furthermore, simple end-to-end scenarios are hereby described: the use of intermediate network entities (like proxy servers, gateways, etc.) is implementation specific and does not alter the scope of the meta-protocol. Actually, one of the peers could eventually be a QoS Broker embedded within the network (case of *active networks*).

VI.1 Sample Scenarios

[0121] Figure 9 presents the three most important scenarios: *association establishment, stream establishment,* and *renegotiation* (in this case, due to a Hand-Over event).

[0122] The session establishment scenario is illustrated in said figure, with respect to the peer B (which, in these scenarios, acts as the callee). As for what peer A (the caller) concerns, the session may have already been established during a previous association establishment with another peer. Nevertheless, as described in § Session Establishment, during each association establishment, session negotiation is performed with respect to the new session participant, by piggybacking Session QoS Profile information.

[0123] Furthermore, during the association establishment phase, session level degradation path information is piggybacked as well.

[0124] In any case, the acknowledgement messages indicate the results of the negotiation process from the callee perspective. If the caller agrees on such results, resources are finally allocated, otherwise corrective actions (which might include interaction with the user) can be taken, in order to reformulate a new *bid*, by using the same message exchanged in the *renegotiation case*. For simplicity reasons, the latter case is not indicated in said figure.

[0125] The following tables describe in detail the methods and the bits they are made of. The notation used is based on the decomposition of methods into information elements, each describing a particular aspect of the negotiation information. Information elements are then further decomposed into tokens, each describing the lowest level of information. Tokens can be used by multiple information elements, as well as the latter can be mapped to multiple methods. The "*m*" symbol indicates a *mandatory* element. The "*o*" symbol indicates an *optional* element. In particular, degradation path information is modeled as additional information elements appended to the reference ones, and linked to the corresponding reference information element as described in § Extended Presentation Context Negotiation Process and depicted in Figure 8.

Table 8:

Tokens	
Name	***Description***
Length	Length of the information element in number of bytes, this token exclusive
Session ID	Globally identifies a session among peers
Association ID	Identifies an association within a given session (cf. SIP Call-ID)
Stream ID	Identifies a stream within a given association
PCI	Identifies the (ASE-related) Presentation Context
Abstract Syntax Name	Identifies the Abstract Syntax
Transfer Syntax Name	Identifies the Transfer Syntax

Table 8: (continued)

Tokens	
Name	***Description***
Component Stub Name	Identifies the Multimedia Component Stub
Component Skeleton Name	Identifies the Multimedia Component Implementation (Skeleton)
Stream QoS Profile	Stream level QoS Profile, described in QML notation
Association QoS Profile	Association level QoS Profile, described in QML notation
Session QoS Profile	Session level QoS Profile, described in QML notation
Application QoS Profile	Application level QoS Profile, described in QML notation
Result	Bitmap indicating the negotiation results: a set of acceptance/provider rejection indicator flags

Table 9:

Information Elements Description	
Name	***Description***
ASE Description	Describes the Presentation Context, augmented with information about any Multimedia Component that might be used by the peers.
Stream Description	Describes the stream-related QoS information
Association Description	Describes the association-related QoS information
Session Description	Describes the session-related QoS information
Application Description	Describes the application-related QoS information
Negotiation Result	Describes the result of the negotiation process

	Length	Session ID	Association ID	Stream ID	PCI	Abstract Syntax Name	Transfer Syntax Name	Component Stub Name	Component Skeleton Name	Stream QoS Profile	Association QoS Profile	Session QoS Profile	Application QoS Profile	Result
ASE Description	m				m	o	o	o	o					
Stream Description	m			m						m				
Association Description	m		m								m			
Session Description	m	m										m		
Application Description	m												m	
Result	m													m

Table 10: Information Elements Structure

Table 11:

Methods Description	
Method Name	***Description***
Association Establishment	Used by the Initiator for negotiating with the Responder the establishment of the given association (and, implicitly, the underlying session), with given QoS.
Stream Establishment	Used by either peer for establishing a new information flow within the given association
Renegotiation	Used by the Initiator for negotiating with the Responder a degradation path for the given association (and, implicitly, the underlying session), given a QoS violation.
Acknowledgement	Reports the result of a (re)negotiation back from the Responder to the Initiator

Table 12:

Methods Structure						
	ASE Description	Stream Description	Association Description	Session Description	Application Description	Result
Association Establishment	m	o	m	m	m	
Stream Establishment	m	m	m	m	m	
Renegotiation	m	o	m	m	m	
Acknowledgement	m	m	m	m	m	m

[0126] The main advantageous differences between the invention and the state of the art

P_ID	Key Aspect	*Description*
1	Interoperability	· Backward compatibility mechanism (see Table 6) · Can operate with compatible QoS proxy disseminated in tree topologies within the network
2	Modularity	· QoS Broker, NRB, Session Manager, Component Chain Coordinator, and Resource Managers are all *components* themselves.
3	Configurability	· NRB can be used for configuring the protocol stack and use any kind of QoS signaling mechanisms · Multimedia Components Skeletons offer multiple implementations of a given contract (Multimedia Component Stub), which can be negotiated as well as Transfer Syntaxes and other OSI Presentation Layer-related issues.
4	Adaptability	· Aggregate Resource Class concept: similar to DiffServ aggregate traffic, but applied to computing unit reosurces. · Fallback mechanism · Piggyback of negotiation information · Piggyback and pre-negotiation of degradation paths · Distributed fallback mechanism with booking mechanism

(continued)

P_ID	Key Aspect	*Description*
5	Standardization	· Formalized as an OSI-like model (differences are the enhancements offered by this invention): more specifically, the hereby-proposed framework focuses on QoS management at a functional level (application, session, association, and stream) level, rather at a structural level (QoS management across protocol stack layers). The former is mapped to the latter by the QoS broker, but is an implementation detail. Therefore the proposed framework is general enough to take into account any type of applications that require QoS guarantees (e.g. this framework can deal also with standalone applications, or applications using inter-processor-communication facilities offered by the Operating System for local communication). · Standard generic solution (meta-protocol) · Implementation independent

Example

[0127] The proposed negotiation protocol can be implemented in the IP world, cast over the Session Initiation Protocol specified by the Internet Engineering Task Force. The procedures, methods, information elements, and tokens described in the previous sections can be in such a case mapped to SIP procedures and methods, modeled according to the SIP text-based message syntax.

[0128] For instance, a possible implementation could be based on the following mapping:

Method	***SIP Method***	***Notes***
Association Establishment	INVITE	Negotiation done during invitation
Stream Establishment	INFO	At the time of this writing, the SIP INFO method is still in an IETF draft state. Among other uses, this method is being used in HMPP for smoothly managing Hand-Over events.
Renegotiation	INFO	At the time of this writing, the SIP INFO method is still in an IETF draft state. Among other uses, this method is being used in HMPP for smoothly managing Hand-Over events.
Acknowledgement	200 OK, INFO	Returning negotiation results to the Initiator, as a result of an invitation. In response to an INFO method, another INFO method shall be used.

[0129] This invention provides the framework for future implementation-specific inventions in the area of application QoS adaptation, especially for mobile and multimedia applications.

[0130] Table 13 captures the most meaningful aspects of this invention, putting in evidence the original ideas being hereby claimed.

Table 13:

Detailed list of the original features of this invention		
P_ID	**Key Aspect**	**Original Features to be claimed**
1	Interoperability	· The fallback ***mechanisms*** described in Table 6 for providing backward compatibility between the various types of applications described in Table 4.
2	Modularity	· The modular ***approach*** for addressing the various types of applications (described in Table 4): the ***staircase*** Figure 2. · The component-based ***model*** applied to both the High-Level QoS API and the Component Model API. See § Modularity.
3	Configurability	· The proposed ***framework*** is general enough to be accommodated on any given *network model* and *QoS signaling protocol*.

Table 13: (continued)

Detailed list of the original features of this invention		
P_ID	**Key Aspect**	**Original Features to be claimed**
		· More specifically, the ***integration*** of the following SW units, which mask implementation details to the QoS middleware and the applications, as described in § Configurability: - *Network Resource Booker (NRB)*: abstracts out network resource reservation mechanisms. It can be directly used by humans (through a proper GUI), or use by the QoS Broker (see Figure 3). - *Session Manager*: coordinates multiple peer to peer associations within a session, and abstracts out session protocols. - *Network Protocol Resource Controller*: abstracts out network protocol stack management (e.g. control of queue scheduler and/or data-link/physical protocol layer resources - through a specific data-link/physical protocol layer manager unit) - *Memory Resource Controller*: abstracts out primary and secondary memory management - *CPU Resource Controller*: abstracts out CPU management (e.g. control over the task scheduler) - Multimedia Device Drivers: abstracts out multimedia device details · See § Configurability.
4	Adaptability	· The ***application*** and ***communication models***, along with their specific adaptation mechanisms, organized in a ***framework.*** - These models are rigorously described in terms of the ISO OSI architecture, and form the logical framework for molding the QoS (re)negotiation process in the most general manner. · The ***High-Level QoS API*** and the ***Component Model API.*** · See § Adaptation Mechanisms.
5	Standardization	· Methods for achieving ***platform-*** and ***network-neutral negotiation*** and ***re-negotiation protocols***, by extensively using ***piggyback mechanisms*** and ***pre-negotiated degradation paths.*** · These results being obtained by introducing the ***concept*** of (re)negotiation process as an extension of the ISO OSI Presentation Negotiation Protocol. See § Proposal of an Standard QoS Negotiation Protocol. · A ***SIP binding*** is offered as an example. This binding to be ***standardized*** through the IETF standardization body. See § Example.

[0131] More specifically, with respect to the Figure 3, this invention proposes an end-terminal QoS middleware architecture.

[0132] Table 14 indicates the dependencies among the SW Units building up said middleware. For a more detailed description of each of these units, please refer to the corresponding entry in Table 14.

Table 14:

List of SW Units	
SW Unit Name	***Dependencies***
QoS Broker	Component lifecycle managed through the Component Coordinator. Coordinates the following units directly: - Network Resource Booker - Session Manager Coordinates the following units through Chain Coordinators: - CPU Manager, - Memory Manager, - Multimedia Component,

Table 14: (continued)

List of SW Units	
SW Unit Name	***Dependencies***
	- Network Protocol Manager, along with their corresponding monitors.
Component Coordinator	Manages lifecycles of the following units: - QoS Broker, - Chain Coordinator, - Session Manager, - Network Resource Booker, - CPU Manager, - Memory Manager, - Multimedia Component, - Network Protocol Manager,
Chain Coordinator	Component lifecycle managed through the Component Coordinator and the QoS Broker. Coordinates the following units: - CPU Manager, - Memory Manager, - Multimedia Component, - Network Protocol Manager,
Session Manager	Component lifecycle managed through the Component Coordinator and the QoS Broker. This Unit offers a unified API with respect to existing session layer protocols, like e.g. SIP or H.323.
Network Resource Booker	Component lifecycle managed through the Component Coordinator and the QoS Broker. Additionally, this unit can be coupled with a GUI, to provide direct human access to its core functionality. This Unit offers a unified API with respect to existing resource reservation protocols/mechanisms, like e.g. IntServ and/or DiffServ.
CPU Manager	Component lifecycle managed through the Component Coordinator and the Chain Coordinator. Associated with a monitor sub-unit, which reports filtered events to said Chain Coordinator. One instance of this unit (and the corresponding monitor sub-unit) is associated with each stream. This unit uses the CPU Resource Controller for getting/changing resource global status information.
Memory Manager	Component lifecycle managed through the Component Coordinator and the Chain Coordinator. Associated with a monitor sub-unit, which reports filtered events to said Chain Coordinator. One instance of this unit (and the corresponding monitor sub-unit) is associated with each stream. This unit uses the Memory Controller for getting/changing resource global status information.
Multimedia Component	Component lifecycle managed through the Component Coordinator and the Chain Coordinator. Associated with a monitor sub-unit, which reports filtered events to said Chain Coordinator. One or more instances of this unit (and the corresponding monitor sub-unit) is associated with each stream. This unit uses the Multimedia Controller for getting/changing resource global status information.
Network Protocol Manager	Component lifecycle managed through the Component Coordinator and the Chain Coordinator. Associated with a monitor sub-unit, which reports filtered events to said Chain Coordinator. One instance of this unit (and the corresponding monitor sub-unit) is associated with each stream. This unit uses the Network Protocol Controller for getting/changing resource global status information.
CPU Resource Controller	Unit providing fine-grained access/control to CPU resource global status. This unit can be embedded into the OS. Used by CPU Managers.

Table 14: (continued)

List of SW Units	
SW Unit Name	***Dependencies***
Memory Controller	Unit providing fine-grained access/control to Memory resource global status. This unit can be embedded into the OS. Used by Memory Managers.
Multimedia Controller	Unit providing fine-grained access/control to Multimedia resources global status. This unit can be embedded into the OS. Used by Multimedia Components.
Network Protocol Controller	Unit providing fine-grained access/control to Network Protocol resource global status. This unit can be embedded into the OS. Used by Network Protocol Managers. This unit can be interfaced to a data link / physical protocol layers managing unit, in order to allow fine parameter tuning and control over said protocol layers from within the QoS management plane. To this extent, the QoS and Mobility Enabled Transport interface indicated in Figure 3 would then redirect QoS-related information/operation to the QoS Management plane and, through the Network Protocol Controller, to the data link / physical protocol layers manager unit. This unit is not explicitly indicated in Figure 3. It is envisioned that the interface between the Network Protocol Controller and the data link / physical protocol layers manager unit will be standardized in the next future.

Claims

1. Processing system for one or more communication networks, providing applications with a platform- and network-independent framework for achieving cross-adaptability by providing components for QoS management in the communication network(s) by means of a component coordinator unit (10).

2. Processing system according to claim 1,
characterised in,
that said generic framework uses a platform- and network-neutral set of application adaptation mechanisms, including a QoS negotiation and re-negotiation protocol.

3. Processing system according to claim 2,
characterised in,
that said protocol uses piggyback mechanisms for QoS negotiating and re-negotiating.

4. Processing system according to claim 1, 2 or 3,
characterised in,
that said generic framework bases on a modular progressive approach to address different types of applications which span from existing application to envisioned more sophisticated applications that rely on middleware for achieving cross-adaptability.

5. Processing system according to one of the claims 1 to 4,
characterised in,
that said generic framework bases on an application model in which each application is allocated to one of a set of application classes having different QoS level with respect to resource usage.

6. Processing system according to claim 5,
characterised in,
that fallback mechanisms are provided for a backward-compatibility between the application classes.

7. Processing system according to claim 5 or 6,
characterised in,
said generic framework bases on a communication model with different functional communication levels for exploiting the various resources in a coordinated manner so as to achieve the desired overall QoS level.

8. Processing system according to claim 7,
characterised in,
that said communication levels include an application (1), a session (2), an association (3) and a stream (4) level.

9. Processing system according to one of the claims 1 to 8,
characterised by
a QoS broker unit (8) being managed by the component coordinator unit (10) and coordinating local and remote resource management by using said negotiation and re-negotiation protocol.

10. Processing system according to claim 9,
characterised by
a network resource booker unit (9) being directly coordinated by the QoS broker unit (8) and managing network resource reservation mechanisms in a implementation independent way.

11. Processing system according to claim 9 or 10,
characterised by
a session manager unit (11) being directly coordinated by the QoS broker unit (8) for establishing and managing sessions in an implementation independent way.

12. Processing system according to claim 11,
characterised by
one or more chain coordinator units (12) being managed by the QoS broker unit (8) through the session manager unit (11) and managing one or more component chains, each chain being associated with a stream.

13. Processing system according to claim 12,
characterised by
one or more CPU-manager units (13) coordinated by the chain coordinator units (12) for managing CPU-resource usage.

14. Processing system according to claim 13,
characterised by
a CPU-resource controller unit (17) providing said CPU-manager units (13) with platform-independent resource status information retrieval and control.

15. Processing system according to one of the claims 12 to 14,
characterised by
one or more memory manager units (14) coordinated by the chain coordinator units (12) for managing memory resource usage.

16. Processing system according to claim 15,
characterised by
a memory controlling unit (18) for providing the memory manager units (14) with platform-independent resource status information retrieval and control.

17. Processing system according to one of the claims 12 to 16,
characterised by
one or more network protocol manager units (15) coordinated by the chain coordinator units (12) for managing network protocol resource usage.

18. Processing system according to claim 17,
characterised by
a network protocol controller unit (19) for providing the network protocol manager units (15) with resource status information retrieval and control.

19. Processing system according to one of the claims 12 to 18,
characterised by
one or more multimedia components (16) coordinated by the chain coordinator units (12) for managing multimedia resources.

20. Processing system according to claim 19,
characterised by
a multimedia controller (20) providing the multimedia component units (16) with platform-independent resource status information retrieval and control.

21. Pieces of software for one or more communication network, being loadable in memory means of one or more notes of the one or more communication networks, providing
applications with a platform- and network-independent framework for achieving cross-adaptability by providing components for QoS management in the communication network(s) by means of a component coordinator unit (10).

22. Pieces of software according to claim 21,
charaterised in,
that said generic framework uses a platform- and network-neutral set of application adaptation mechanisms including a QoS negotiation and re-negotiation protocol.

23. Pieces of software according to claim 22,
characterised in,
that said protocol uses piggyback mechanisms for QoS negotiating and re-negotiating.

24. Pieces of software according to claim 21, 22 or 23,
characterised in,
that said generic framework bases on a modular progressive approach to address different types of applications which span from existing applications to envisioned more sophisticated applications that rely on middleware for achieving cross-adaptability.

25. Pieces of software according to one of the claims 21 to 24,
characterised in,
that said generic framework bases on an application model in which each application is allocated to one of a set of application classes having different QoS level with respect to resource usage.

26. Pieces of software according to claim 25,
characterised in,
that fallback mechanisms are provided for a backward-compatibility between the application classes.

27. Pieces of software according to claim 25 or 26,
characterised in,
said generic framework bases on a communication model with different functional communication levels for exploiting the various resources in a coordinated manner so as to achieve the desired overall QoS level.

28. Pieces of software according to claim 27,
characterised in,
that said communication levels include an application (1), a session (2), an association (3) and a stream (4) level.

29. Pieces of software according to one of the claims 22 to 28,
characterised by
a QoS broker unit (8) being managed by the component coordinator unit (10) and
coordinating local and remote resource management by using said negotiation and re-negotiation protocol.

30. Pieces of software according to claim 29,
characterised by
a network resource booker unit (9) being directly coordinated by the QoS broker unit (8) and managing network resource reservation mechanisms in an implementation independent way.

31. Pieces of software according to claim 29 or 30,
characterised by
a session manager unit (11) being directly coordinated by the QoS broker unit (8) for establishing and managing sessions in an implementation independent way.

32. Pieces of software according to claim 31,
characterised by
one or more chain coordinator units (12) being managed by the QoS broker unit (8) through the session manager unit (11) and managing one or more component chains, each chain being associated with a stream.

33. Pieces of software according to claim 32,
characterised by
one or more CPU-manager units (13) coordinated by the chain coordinator units (12) for managing CPU-resource usage.

34. Pieces of software according to claim 33,
characterised by
a CPU-resource controller unit (17) providing said CPU-manager units (13) with platform-independent resource status information retrieval and control.

35. Pieces of software according to one of the claims 32 to 34,
characterised by
one or more memory manager units (14) coordinated by the chain coordinator units (12) for managing memory resource usage.

36. Pieces of software according to claim 35,
characterised by
a memory controlling unit (18) for providing the memory manager units (14) with platform-independent resource status information retrieval and control.

37. Pieces of software according to one of the claims 32 to 36,
characterised by
one or more network protocol manager units (15) coordinated by the chain coordinator units (12) for managing network protocol resource usage.

38. Pieces of software according to claim 37,
characterised by
a network protocol controller unit (19) for providing the network protocol manager units (15) with resource status information retrieval and control.

39. Pieces of software according to one of the claims 32 to 38,
characterised by
one or more multimedia components (16) coordinated by the chain coordinator units (12) for managing multimedia resources.

40. Pieces of software according to claim 39,
characterised by
a multimedia controller (20) providing the multimedia component units (16) with platform-independent resource status information retrieval and control.

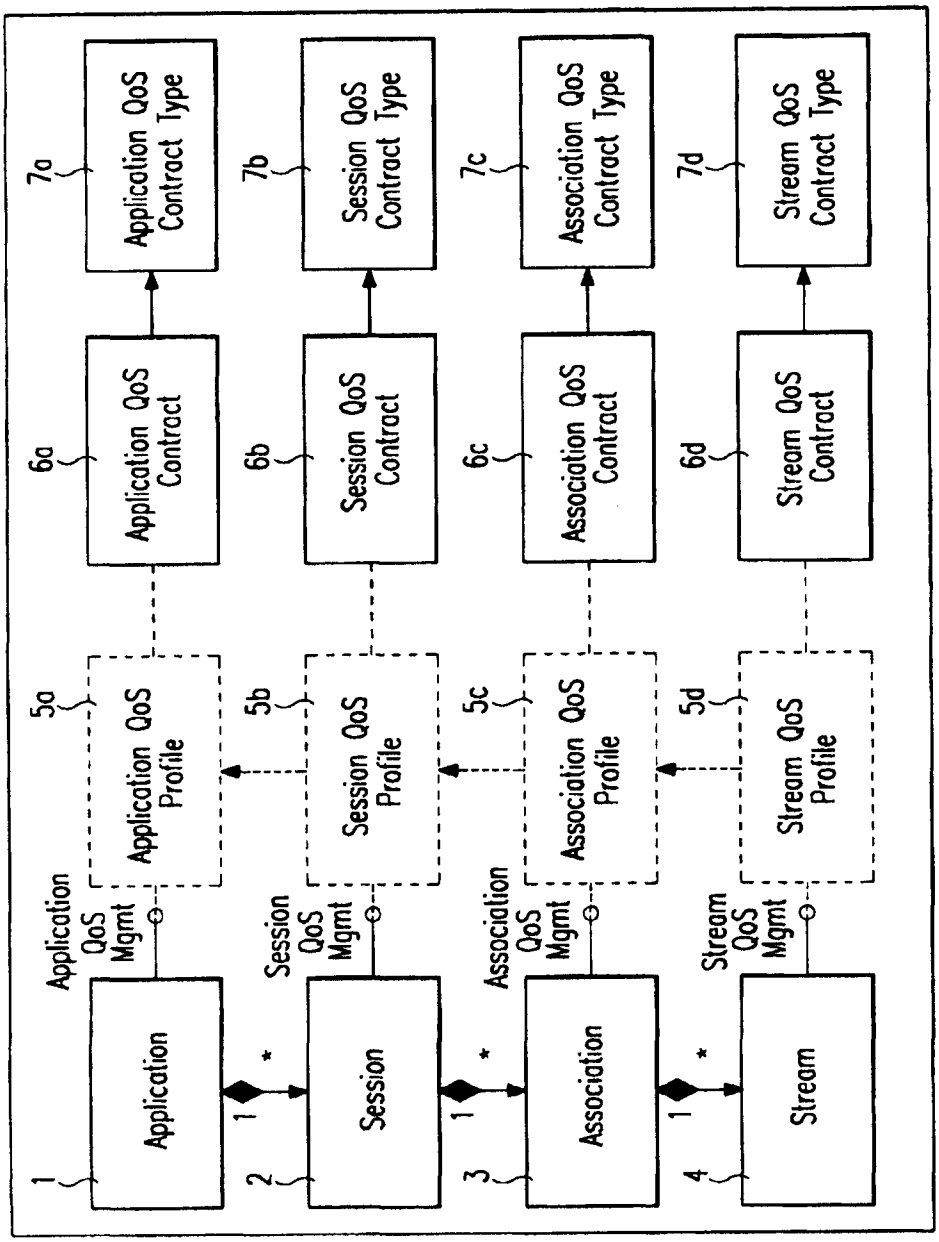

QoS Adaption Framework
Application
Session
Association
Stream
Application QoS Mgmt
Session QoS Mgmt
Association QoS Mgmt
Stream QoS Mgmt
Application QoS Profile
Session QoS Profile
Association QoS Profile
Stream QoS Profile
Application QoS Contract
Session QoS Contract
Association QoS Contract
Stream QoS Contract
Application QoS Contract Type
Session QoS Contract Type
Association QoS Contract Type
Stream QoS Contract Type
1
2
3
4
5a
5b
5c
5d
6a
6b
6c
6d
7a
7b
7c
7d
*
1

Fig. 1

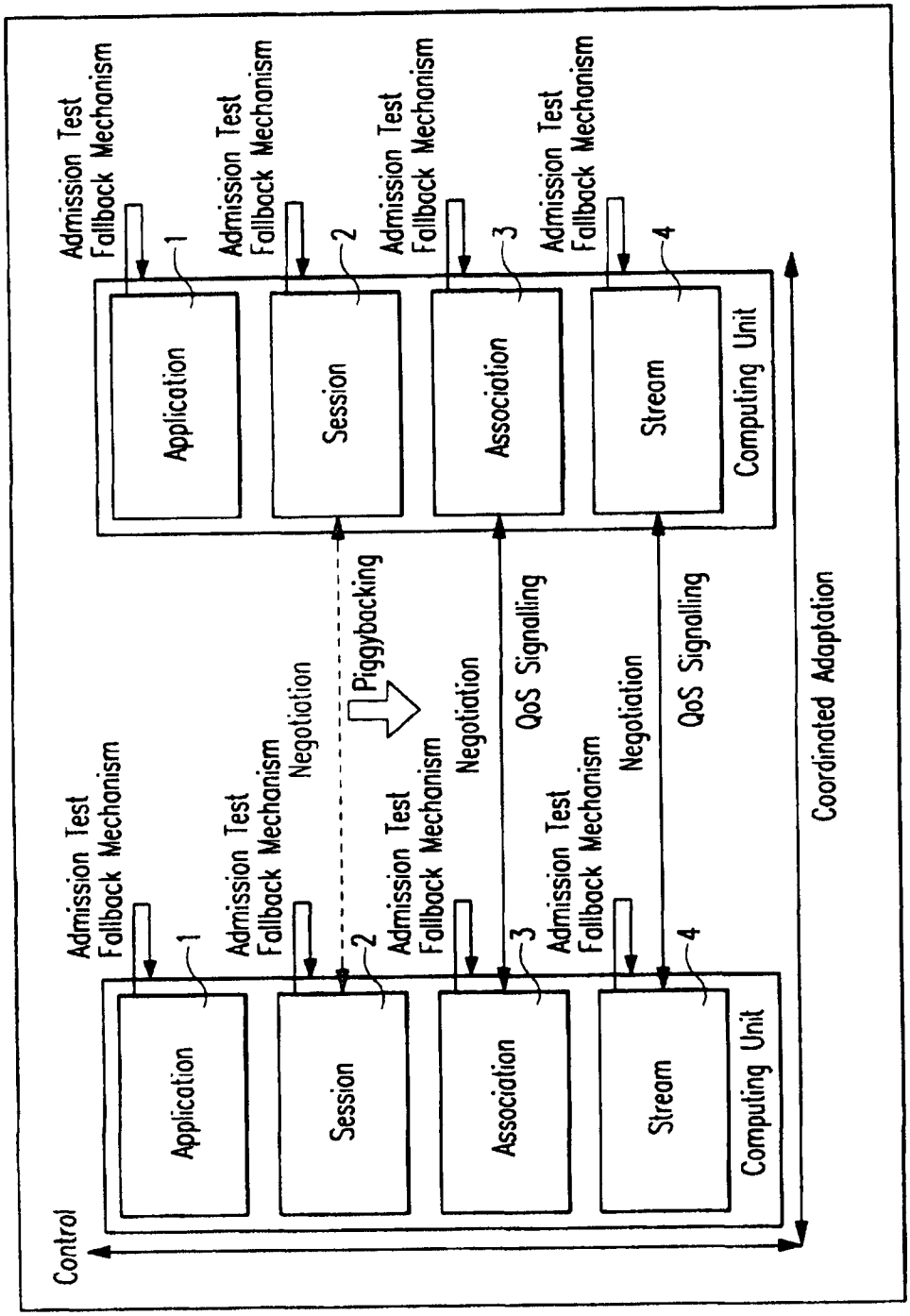

Admission Test
Fallback Mechanism
1
Application
2
Session
3
Association
4
Stream
Computing Unit
Coordinated Adaptation
Piggybacking
Negotiation
QoS Signalling
Control
Coordinated Adaptation vs. Control mechanisms

Fig. 2

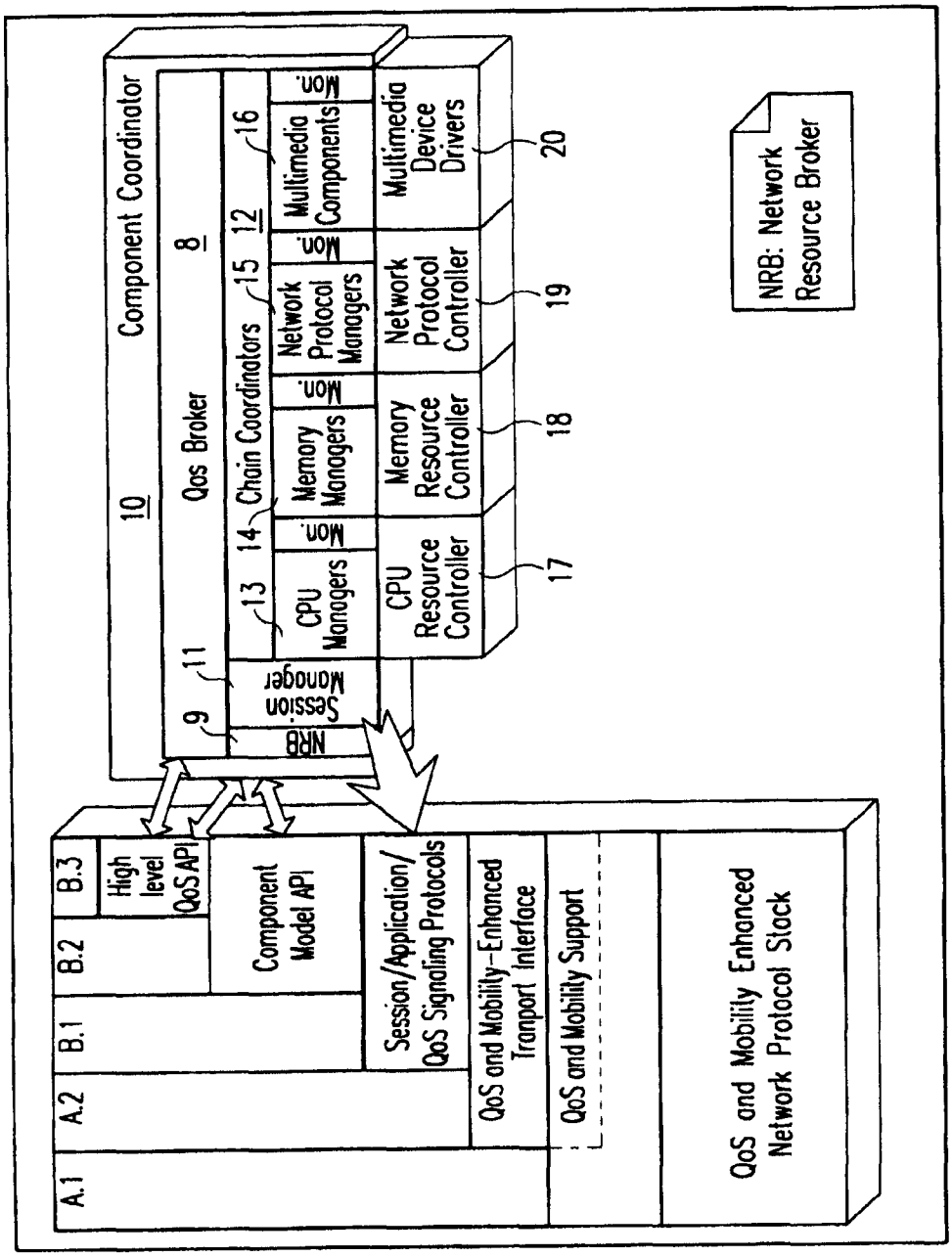

Component Coordinator
10
Qos Broker
8
9
NRB
11
Session Manager
13
CPU Managers
14
Chain Coordinators
15
Memory Managers
Network Protocol Managers
12
16
Mon.
Multimedia Components
CPU Resource Controller
17
Memory Resource Controller
18
Network Protocol Controller
19
Multimedia Device Drivers
20
NRB: Network Resource Broker
A.1
A.2
B.1
B.2
B.3
High level QoS API
Component Model API
Session/Application/ QoS Signaling Protocols
QoS and Mobility-Enhanced Tranport Interface
QoS and Mobility Support
QoS and Mobility Enhanced Network Protocol Stack
End System Reference Architecture

Fig. 3

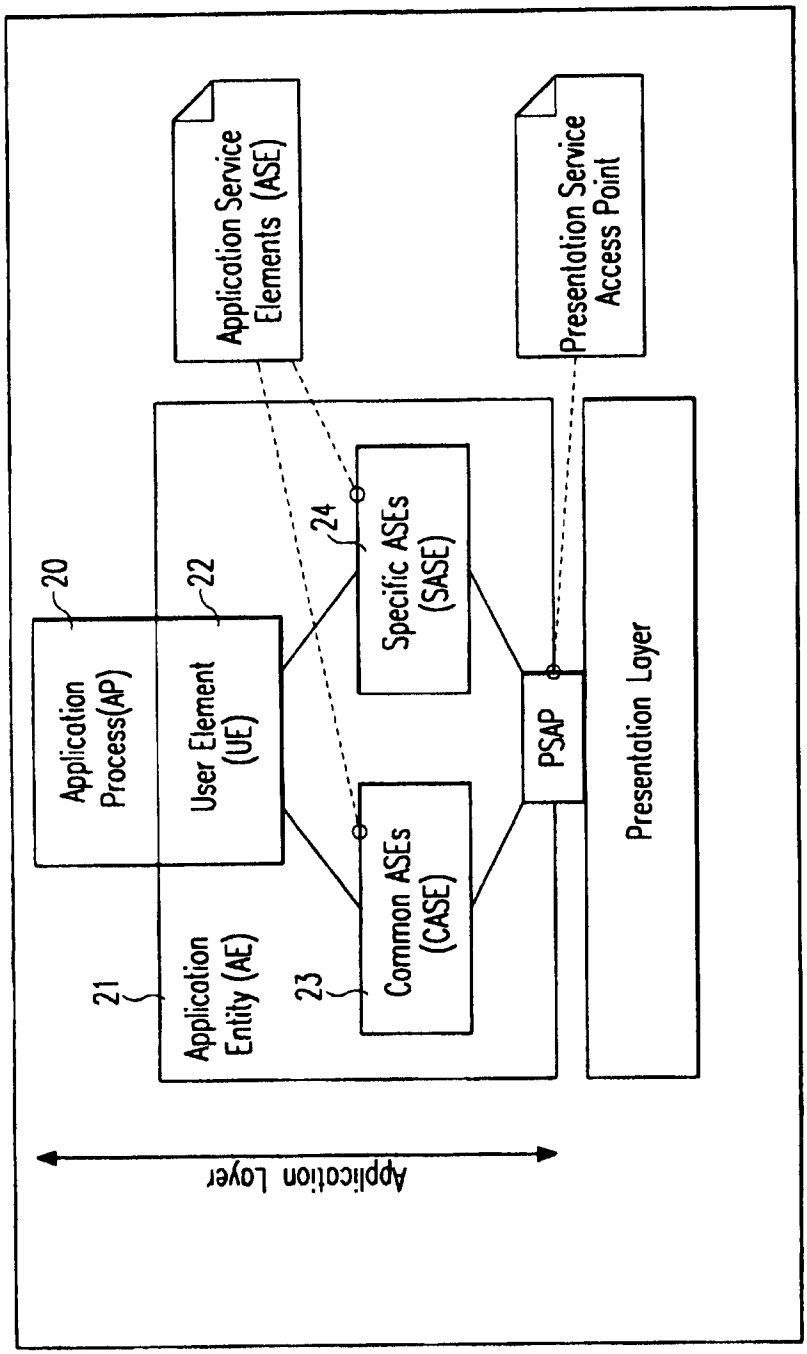
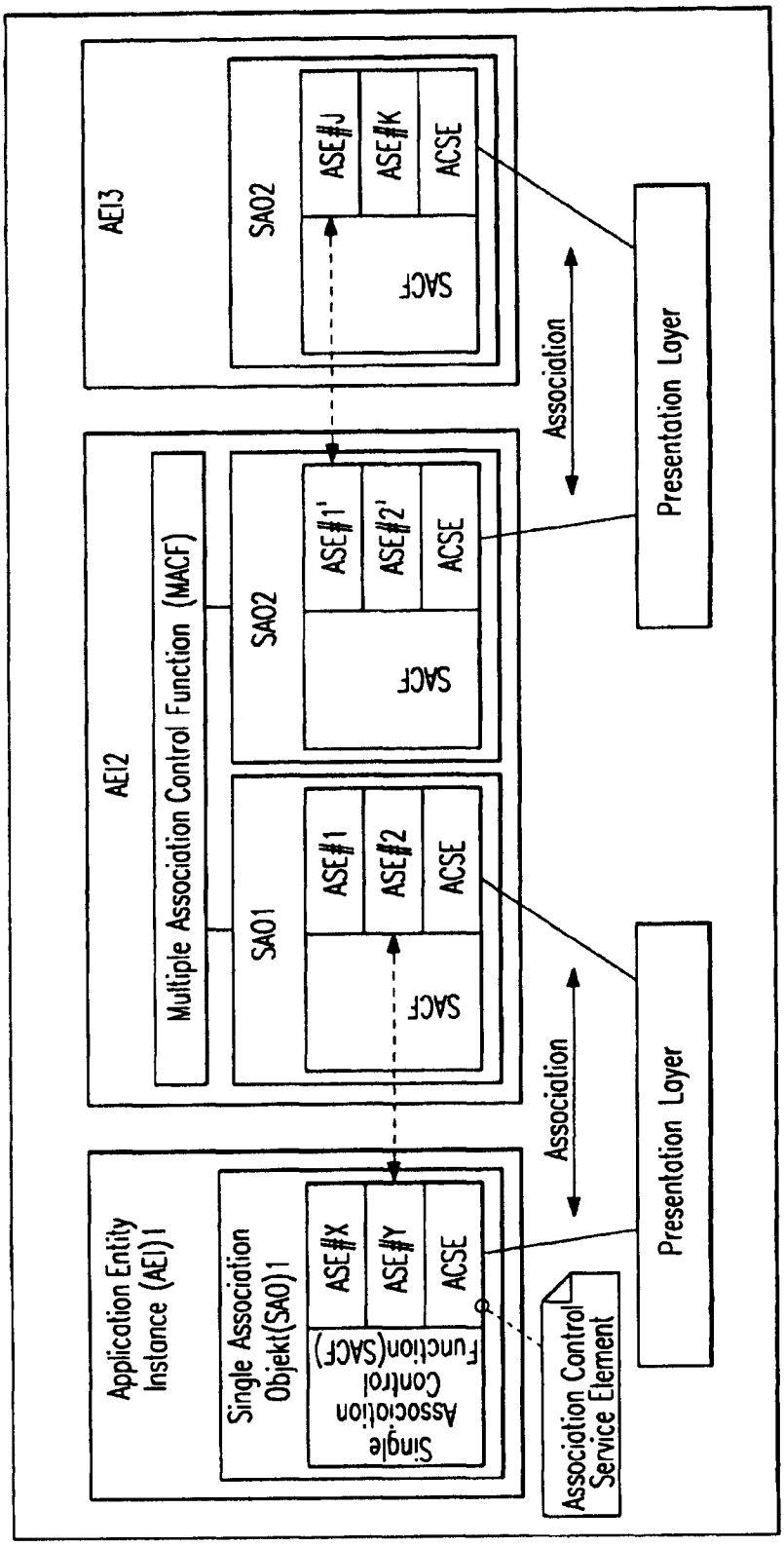
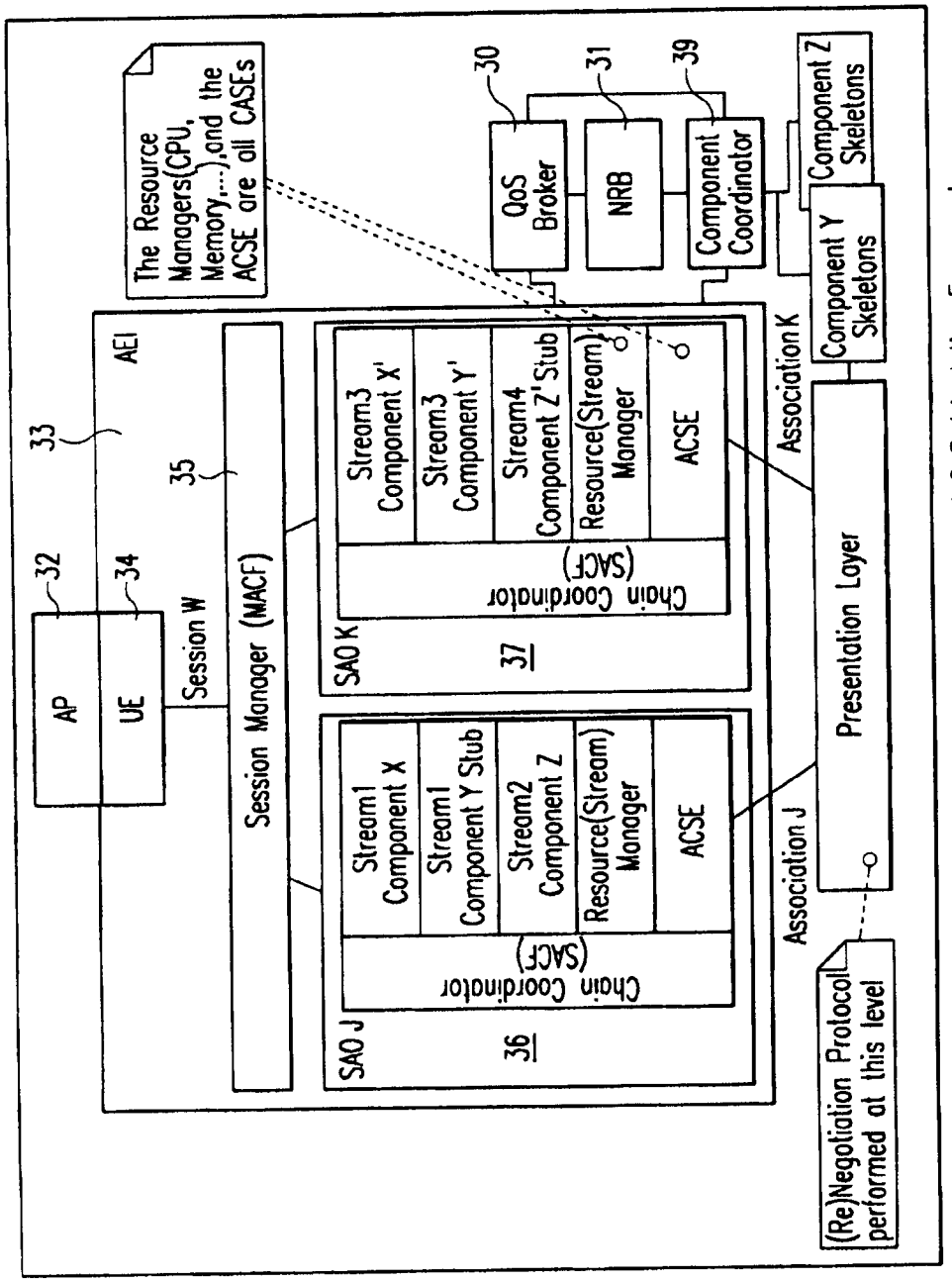

Application Service Elements (ASE)
Presentation Service Access Point
20
22
24
Application Process(AP)
User Element (UE)
Specific ASEs (SASE)
PSAP
Presentation Layer
21
23
Application Entity (AE)
Common ASEs (CASE)
Application Layer
The OSI Application Layer

Fig. 4



Example of single and multiple association

Fig. 5



A formal description of the proposed QoS Adaptation Framework

Fig. 6

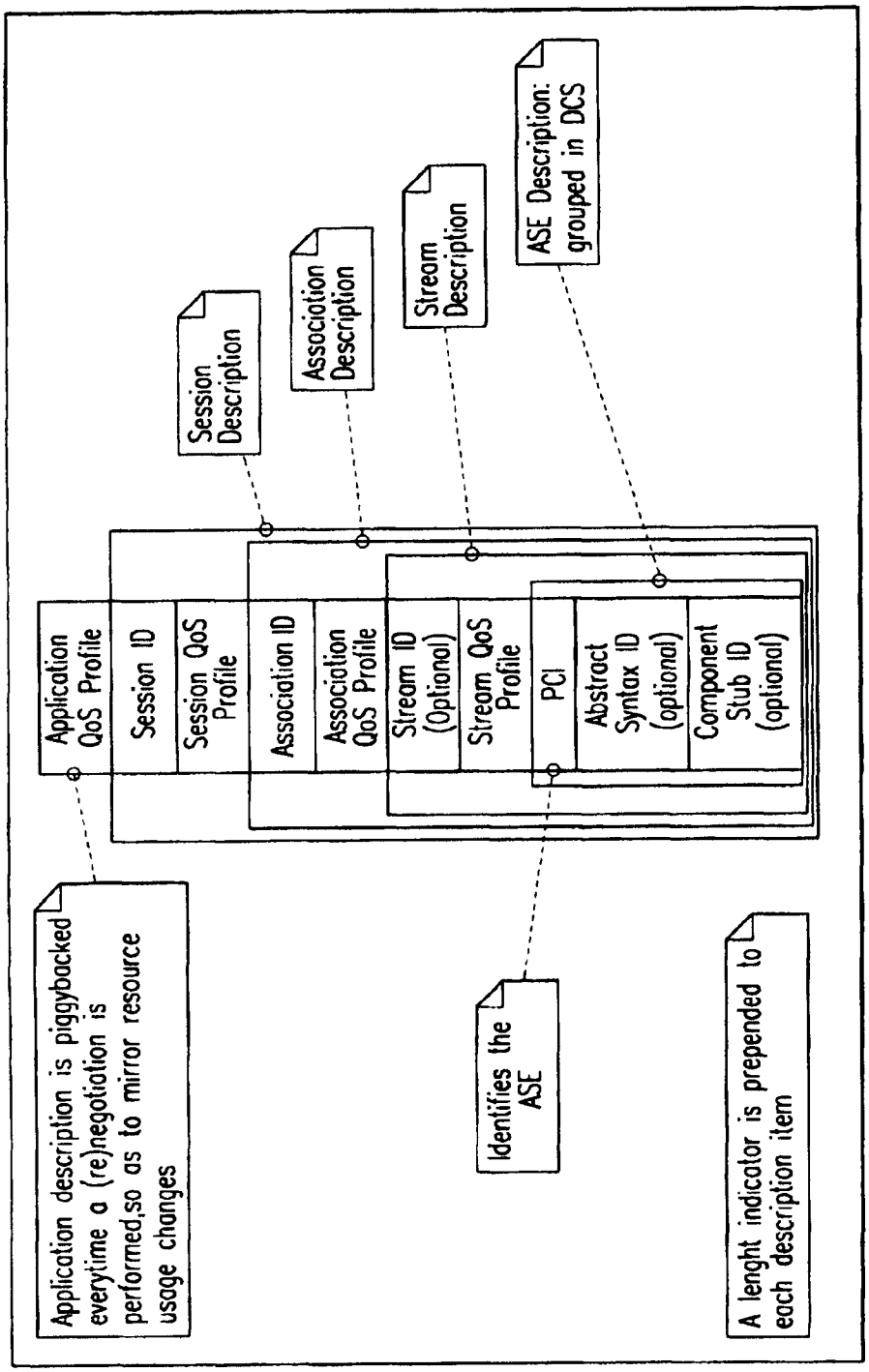

Application description is piggybacked everytime a (re)negotiation is performed,so as to mirror resource usage changes
Application QoS Profile
Session ID
Session QoS Profile
Association ID
Association QoS Profile
Stream ID (Optional)
Stream QoS Profile
PCI
Abstract Syntax ID (optional)
Component Stub ID (optional)
Session Description
Association Description
Stream Description
ASE Description: grouped in DCS
Identifies the ASE
A lenght indicator is prepended to each description item
Logical structure of the negotiated information

Fig. 7

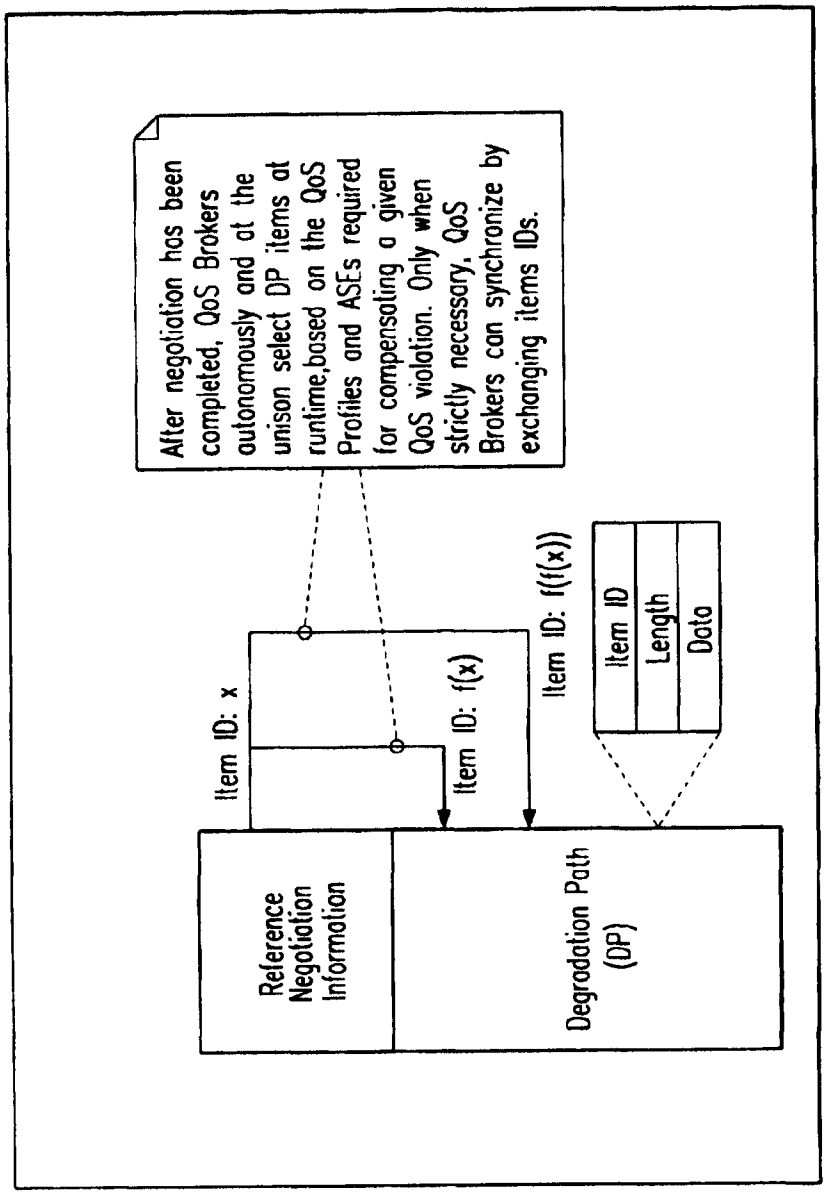

After negotiation has been completed, QoS Brokers autonomously and at the unison select DP items at runtime,based on the QoS Profiles and ASEs required for compensating a given QoS violation. Only when strictly necessary, QoS Brokers can synchronize by exchanging items IDs.
Item ID: x
Item ID: f(x)
Item ID: f(f(x))
Item ID
Length
Data
Reference Negotiation Information
Degradation Path (DP)
Degradation Path information exchanged during negotiation

Fig. 8

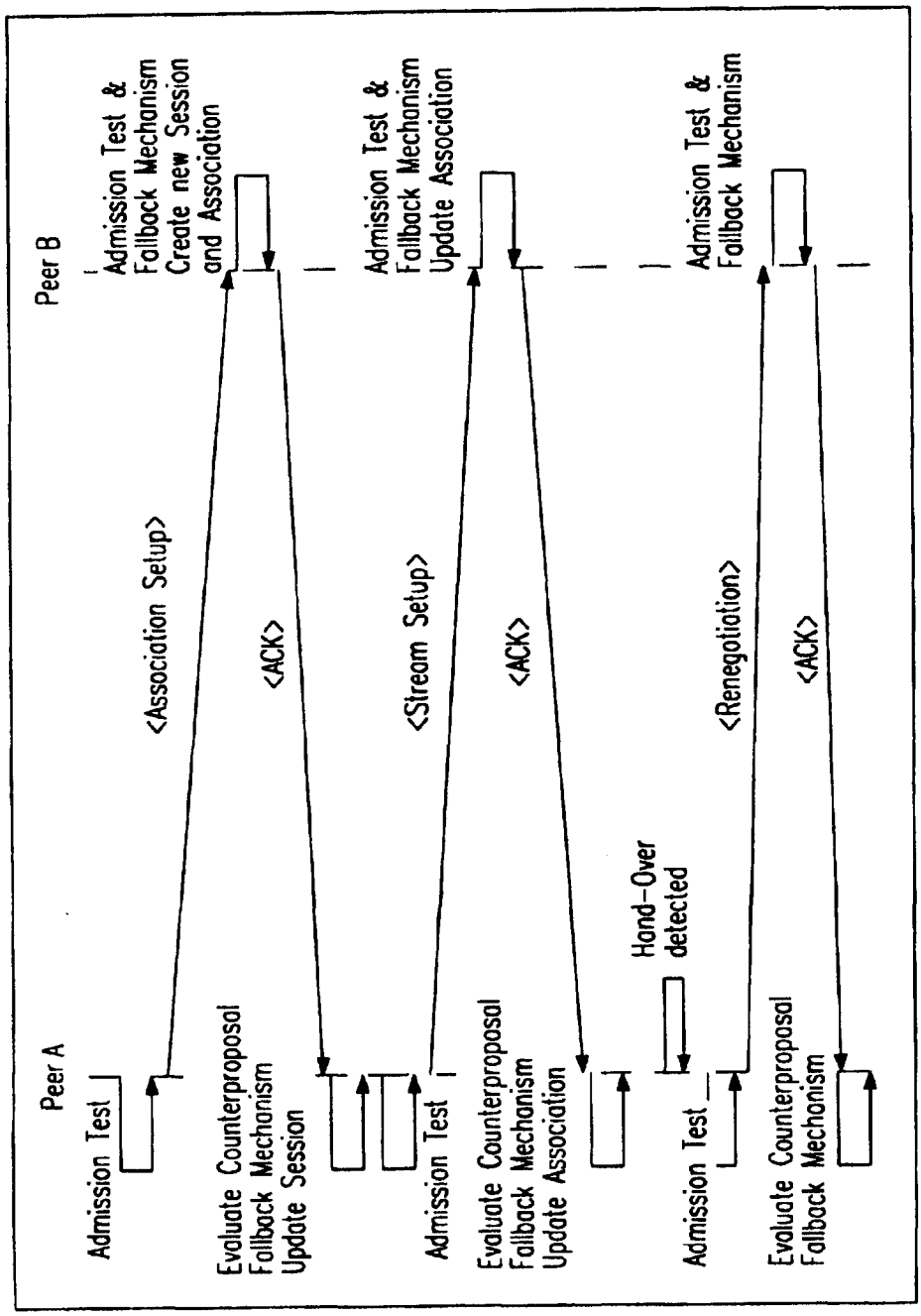

Peer A
Peer B
Admission Test
Admission Test & Fallback Mechanism Create new Session and Association
<Association Setup>
Evaluate Counterproposal Fallback Mechanism Update Session
<ACK>
Admission Test
Admission Test & Fallback Mechanism Update Association
<Stream Setup>
Evaluate Counterproposal Fallback Mechanism Update Association
<ACK>
Hand-Over detected
Admission Test
Admission Test & Fallback Mechanism
<Renegotiation>
Evaluate Counterproposal Fallback Mechanism
<ACK>
Association/Stream Set-up and HAND-Over scenarios

Fig. 9

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 11 1191

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	BARZILAI T P ET AL: "DESIGN AND IMPLEMENTATION OF AN RSVP-BASED QUALITY OF SERVICE ARCHITECTURE FOR AN INTEGRATED SERVICES INTERNET" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS,US,IEEE INC. NEW YORK, vol. 16, no. 3, 1 April 1998 (1998-04-01), pages 397-413, XP000740059 ISSN: 0733-8716 * paragraph '0001! * * paragraph '0011! * * paragraph '0111! *	1,4-21, 24-40	H04L29/06 H04L12/56
Y		2,3,22, 23	
Y	EP 0 806 855 A (SUN MICROSYSTEMS INC) 12 November 1997 (1997-11-12) * abstract * * page 2, line 33 - page 3, line 18 * * figure 3 *	2,3,22, 23	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 January 2001	Canosa Aresté, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 00 11 1191

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

29-01-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0806855 A	12-11-1997	US 5708778 A	13-01-1998
		JP 10084399 A	31-03-1998